
# A Blockchain solution to Electronic Prescriptions and Electronic Prescribing in Ireland

## Final Year Project Report

**TU856**
**BSc in Computer Science**

**Daniel Simons**

**C17371946**

**Luis Miralles**

School of Computer Science

Technological University, Dublin

**05/04/2021**

# Abstract

The goal of this project was to create a secure immutable ledger which would securely store prescription information using blockchain technology. This blockchain network can be accessed through a web-based application and allow service users to view and send their prescriptions while medical practitioners can send prescriptions to service users.

Blockchain technology will address the problem of lost or stolen prescriptions and prevent the risk of data breaches or data leaks. The proposed system proved to reduce errors made in the dispensing stage of prescriptions by reducing manual data entry and retrieval. The Health Information and Quality Authority (HIQA) has researched the benefits of electronic prescribing (ePrescribing) and found benefits to the service user. This research was born from the acknowledgement that medication errors from prescription through to dispensation may be addressed by introducing technology solutions (1).

Blockchain technology is accessible by anyone who has their private key and can access and control the prescriptions prescribed by an authorized medical practitioner. Pharmacies are aware of the medication required which enable them to process the receipt of prescription renewals more efficiently and effectively. The benefit to the service user is the assurance that their data is secure and private.

Blockchain technology will reduce the reliance on less reliable, traditional means of information sharing. Australia and the United Kingdom are currently using a similar less secure version of ePrescribing with great success (2). The author has created a prototype system that could be considered as Ireland's healthcare evolves and incorporates an ePrescription system approach. The evaluation of the effectiveness of this proposal is done through unit and systems testing together with integration testing. The proposal was underpinned by the findings and legislation from HIQA and the Pharmaceutical Society of Ireland (PSI) (3). The project was trialed with user acceptance testing inclusive of users of different ages and capabilities.

# Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

_____

Daniel Simons

05/04/2021

# Acknowledgements

I would like to express my gratitude towards my supervisor, Luis Miralles, who has been a great help with this dissertation.

I would like to thank my mother Rachel, who has helped me focus and keep motivated over the last couple of weeks while also helping me by reading over my work and making sure that I made sense, thank you so much Mum.

I would like to thank the rest of my family also; they have kept me positive throughout the course of my Dissertation.

I would like to thank my partner Dearbhla, who has provided me with continuous support and tolerates my high stress levels and gives me nothing but love and patience, thank you Derv.

Finally, I would like to thank Technological University Dublin for their guidance, support and teaching. I would not have made it to this year without it.

# Table of Contents

# Table of figures

# List of tables

# 1. Introduction

## 1.1. Preamble and the impact of Covid-19

Throughout the Covid-19 pandemic, General Practitioners (GP), clinics and hospitals experienced an unprecedented rise in workload alongside the need to maintain social distancing. This presented a significant problem for service users who were and continue to be encouraged to not attend their GPs in person. The paper-based prescription system has been a particular challenge that required an immediate change in the process to enable service users to access their medications.

In response to the Covid-19 outbreak in Ireland, temporary amendments to the medicinal products regulations and the misuse of drugs regulations were made to allow the electronic transfer of prescriptions using a service called Healthmail (4). Healthmail provided some improvements to the paper-based prescription system such as improving the speed at which medical information could be transferred (5) and reducing the risk of stolen or missing prescriptions. There are some drawbacks to Healthmail such as service users' data is not in their control but instead in the control of a medical practitioner and there remains a reluctance from some medical workers to use it (6). More of the benefits and disadvantages of Healthmail will be discussed in Section 2.3.1.

There have been many other electronic prescription systems or services deployed internationally. Most of these ePrescribing systems improved upon the paper-based prescription system drastically by reducing prescription errors and allowing for quicker transmission of data but they also have their drawbacks. Most of these systems have security issues and or the inability for service users to access their own prescription records (cf. 2.3.).

An electronic prescription system similar to this project is the Decentralized Blockchain Ledger for the Management of Medication Histories (DMMS) (7). This is a blockchain application for ePrescribing which is very similar to how this system operates. This application had many benefits, which includes advanced security for the network. There are some slight improvements that could be made such as a shared system with the emergency department so that all records can be sent between two establishments. The ability to remove the service user's data once requested by the user and not kept permanently on the ledger would also attend to the rights of the service user in line with General Data Protection Regulations (GDPR) and the right to be forgotten (8).

These benefits and disadvantages were considered when designing this application to attend to this gap. This blockchain application will eliminate some of the issues encountered with ePrescribing systems that have come before.

## 1.2. Blockchain Technology

Blockchain Technology provides a secure decentralized ledger where each transaction between the service user, GP and pharmacist is logged and immutable to provide a clear history and understanding of what prescription exchange took place. This application provides a secure way of ePrescribing that will not be the subject of phishing or vulnerable to distributed denial of service (DDoS) attacks. The application will be a web application that allows for easy access to service users, GPs, and pharmacies. Service users are in control of their prescriptions with either a mobile application or on their private computers (PCs) in line with GDPR (8).

ePrescribing can be used to alleviate errors previously found in paper-based prescription by removing manual data entry and can send a prescription in advance to a pharmacy. Medication can then be collected from a pharmacy of choice using a request feature, and to renew a prescription with a simple command.

Another aspect of this project was to make sure it complies with the ePrescribing standards provided by HIQA and other ePrescribing services which are underpinned by the legislation from the Pharmaceutical Society of Ireland (PSI).

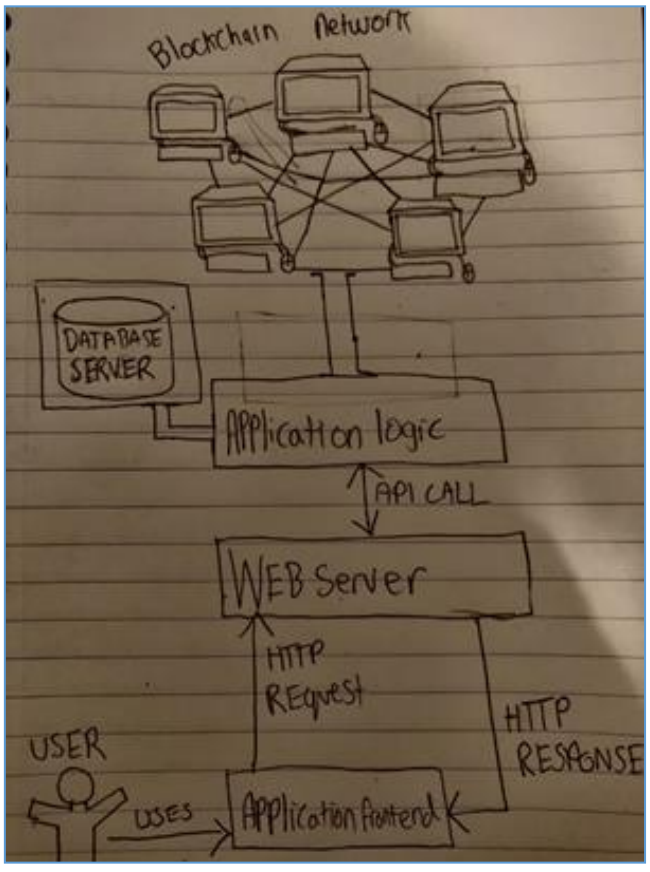



*Figure 1 - Draft Proof of Concept*

## 1.3. Project Aims and Objectives

The overall objective of this project was to develop a secure, easy to use blockchain web application that provides a reliable system to create, send, and receive medical prescriptions. Blockchain Technology was designed to encompass people of all ages and who have varying technical abilities and is accessible from any device with an internet connection.

The application was designed to abide by pharmaceutical prescription legislation and GDPR and to improve upon previous ePrescribing systems that were reported to have had critical flaws.

The following actions were central to assist the author in achieving the project aims and objectives:

- Understand and research how the current ePrescription service in Ireland operates and learn about other ePrescribing systems around the world.
- Understand and research how blockchain technology works and how it helped improve ePrescribing in Ireland.
- Research web applications that use blockchain.
- Research GDPR and other pharmaceutical legislation that underpinned this project.
- Design a 3-tiered web application that uses blockchain technology.
- Develop a web application with blockchain technology which improves security and efficiency for an ePrescribing service.
- Implement user acceptance tests to ensure that the application is easy to use for all ages and skill levels.
- Prove that a blockchain application can be used as an ePrescribing system.
- Prove that prescription errors and lost prescriptions can be prevented with the use of this system.

## 1.4. Project Scope

The project aims to create a blockchain-based prescribing application. However, the project did not explore the technicalities of recommending or evaluating medication chosen or prescribed to a patient. The examination of prescribed medications requires medical qualifications, experience and a forensic knowledge of pharmaceuticals which is outside the scope of this project. Blockchain Technology will simply be a medium for ePrescribing to occur.

## 1.5. Thesis Roadmap

- Literature review

  This section explored the background research into ePrescribing today in Ireland and around the world, the issues surrounding it, and how blockchain technology can help resolve these issues. The author determined suitable blockchain technology solutions and requirements needed for a secure ePrescribing service.

- Prototype Design

  The design chapter explored the methodology and the approach taken for the project. It explored the technical architecture of the program and the designs such as use case diagrams, class diagrams, entity relational diagrams which were needed to help aid the development of this project.

- Prototype Development

  The prototype development chapter discussed the implementation of the proposed design and elaborates on the choices taken in the design phase and highlights any changes that were made to the design.

- Testing and Evaluation

  This section provides an analysis of the different types of testing that were used throughout the project. Unit, systems, and user acceptance testing are just some of the tests that were used. Additionally, the evaluation of the program that was undertaken was also discussed.

- Issues and Future Work

  In this chapter the key points from this project were discussed, along with the challenges and the knowledge gained. It also outlined the successes and improvements that could be made in the future.

# 2. Literature Review

## 2.1. Introduction

This chapter reviews the relevant research and software for the scope of this dissertation. The research will be conducted in accordance with the research strategy provided (cf. 2.1.1) and the key findings will be discussed. The research will be funnelled down into four main sections which will be:

- Background Research into current issues with prescriptions and ePrescribing.
- Existing Solutions which function similarly to this project and resolve issues equivalently to this project.
- Technologies researched in relation to developing this project such as relevant blockchain frameworks and test harnesses.
- Other research such as blockchain technology and technical standards in this area including two final year projects.

### 2.1.1 Research strategy

The research was conducted using the facilities available at the Technology University of Dublin which were accessed remotely. The search engine used primarily in this dissertation is Google and Google scholar. The material accessed through this search engine was mainly a combination of academic papers, publications, and whitepapers along with standard web information.

The search terms that were used to find relevant information to this project were as follows; Electronic Health Records, Prescriptions, ePrescriptions, ePrescribing, Blockchain, Blockchain infrastructure, Blockchain, and healthcare. Materials were examined to ensure that the most recent and relevant information was used. Materials were identified by initially reading the abstract online and checking if they included the relevant search terms. Research that was found to be relevant was added to an appropriate excel spreadsheet for further analysis. In this spreadsheet, the link, heading, and keywords associated with the material was captured which facilitated easy access and retrieval of information.

### 2.1.2 Interview structure

The interviews were conducted using a semi-structured approach. Semi-structured interviews are designed to retrieve information from persons of relevant experience which captures the scope of their opinions and knowledge in the area.

The questions are styled to be open-ended so that the respondent can give a more detailed response with details that the interviewer may not be aware of (9).

The questions were designed to be cognisant of the scope of blockchain technology and the aims and objectives of the study. The questions were broadly worded to capture the full spectrum of their experiences and to ensure no bias influenced their decisions or opinions.

## 2.2. Background Research

Ireland's current paper-based prescription system has its flaws and limitations. 27% of prescriptions have been found to be the cause of errors because of the illegibility of the handwriting which obscures critical medical information needed (10). There are also many other issues with paper-based prescription systems such as lost or stolen prescriptions, time-consuming aspects of writing and searching for prescriptions, and the breaches of privacy that can occur. These flaws were only highlighted more during the recent Covid-19 pandemic when hospitals and pharmacies were overwhelmed by the virus. This has presented a significant problem for service users who are and continue to be encouraged to not attend their GPs in person. The paper-based prescription system has been a particular challenge that required an immediate change in the process to enable service users to access their medications.

In response to the Covid-19 outbreak in Ireland, temporary amendments to the medicinal products regulations and the misuse of drugs regulations were made to allow the electronic transfer of prescriptions using a service called Healthmail (4). Before this, it was a legal requirement to produce a paper prescription for service users to present to pharmacists. This was a step in the right direction for prescribing in Ireland but there are still some flaws that will be discussed further (cf. 2.3).

## 2.3. Alternative Existing Solutions to Your Problem

This section discusses the advantages and disadvantages of current solutions to ePrescribing nationally and internationally. These alternative solutions were closely analysed when designing this proposed solution.

### 2.3.1 Healthmail

Healthmail (11) is an email service that allows the transfer of medical information between a variety of medical workers. Healthmail was created in 2014 and the number of registered accounts has slowly increased to over 2000 as of December 2017(12).

Healthmail is a portal which can be accessed by medical practitioners using a dedicated email address and password (cf. Figure 2). Healthmail provides many benefits over the old system of posting or faxing prescriptions. It provides a safe manner to send private data and provides a quicker way of sending and receiving information (5). This information is sent through transport layer security (TLS) (13) which encrypts information or data in transit, this prevents the unwarranted interception of patient data or sensitive data. For prescriptions,

this information is usually an email with an image of the prescription as an attachment. Healthmail allows storage of up to 2GB of data per Healthmail user (11). Many GPs speak positively about Healthmail and believe that it “improves patient care” and that the service should continue (6).


*Figure 2 - Image from Healthmail Login Page*

There are many benefits to the Healthmail system over the paper-based system but there are also some disadvantages to the system as well.

Healthmail is only available to medical workers which means service user’s data is in other parties’ control. There remains a reluctance from medical workers to use the Healthmail service with only 2,367 using the service since December 2017(12), however, this figure has been increasing incrementally over the last number of years as shown in Figure 3.

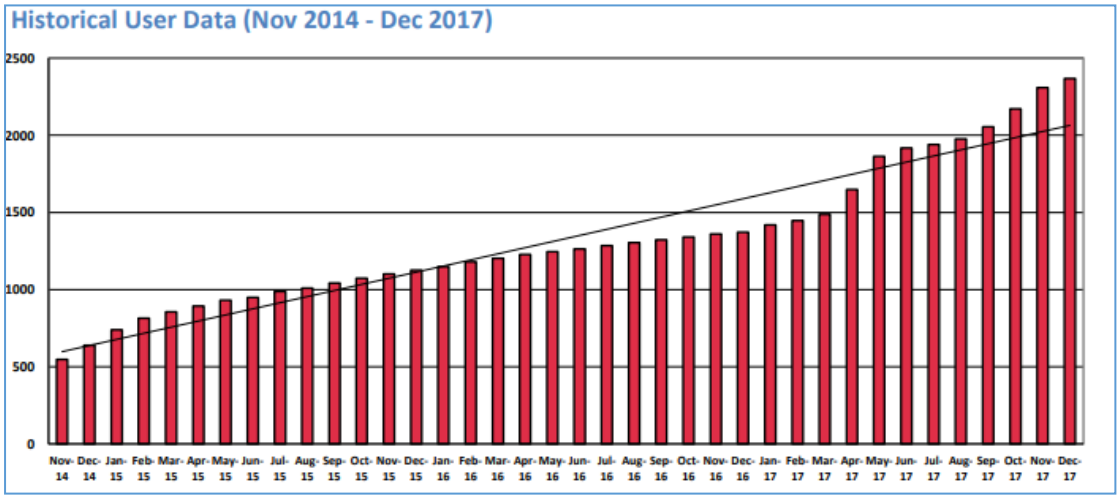



*Figure 3 - Historical User Data (November 2014 - December 2017)*

Another issue found with the Healthmail service is the degree of uncertainty that emails are read. This issue caused Healthmail users to turn to alternative less secure ways of communication (5). The issue of illegible handwriting on prescriptions is not dealt with in

this approach as a capture of the prescription is taken rather than typed out or digitized for readability.

Emails are also subject to phishing attacks; phishing attacks are an attack method where websites are spoofed to harvest information (14). These links are commonly shared by email (15). It is estimated that in April 2020, 18 million phishing emails related to Covid-19 were sent out and that this year phishing attacks have increased by 32% (16).

These issues were taken into consideration when designing the blockchain technology and mitigated the security risks associated with Healthmail or other email-based services; whilst also providing confirmation of the information in a clear and recognizable way.

While Healthmail provided a step in the right direction for ePrescribing in Ireland, it is not a strictly ePrescribing service but rather a service that allows the transfer of medication information. The following examples of ePrescribing from an international perspective will also provide insight into the evolution of ePrescribing. The author included each company's logo, as for some companies, their logo is more recognisable than their name.

### 2.3.2 MediSecure & eRx (Australia)

There are roughly 200 million prescriptions each year that are sent through Australia's ePrescribing system (17) as seen in Figure 4. Australia's ePrescription system is controlled by two companies, MediSecure and eRx. These two companies operate in a similar way with both operating a nationally linked database where service users can manage their own information including prescriptions. Both services can be broken down into similar steps, the doctor or GP must first create the prescription and print it out for the service user, this prescription contains a barcode to allow the patient access to medication from a pharmacist. This barcode is linked to an eScript which is stored in a linked database. This barcode can also be sent to a mobile device as seen in Figure 5.

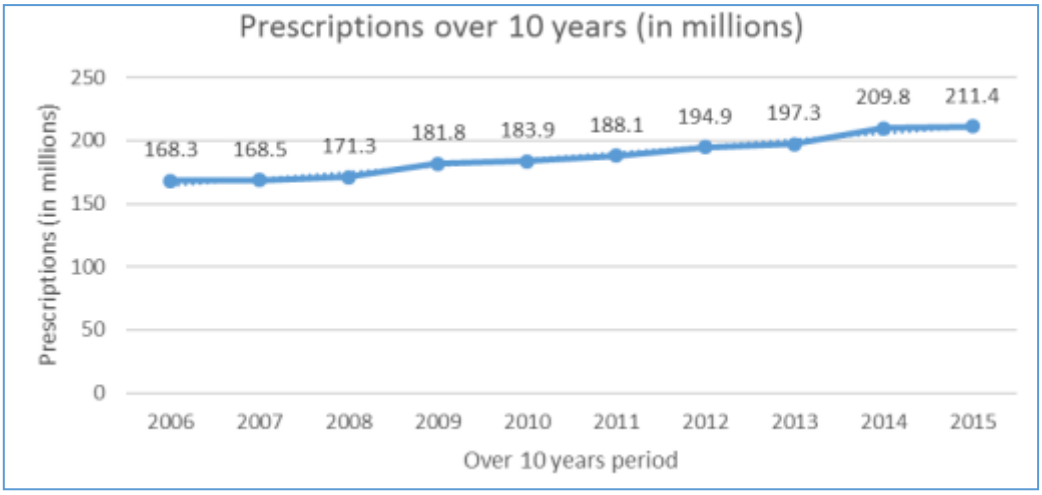



*Figure 4 - Number of Prescriptions Sent*

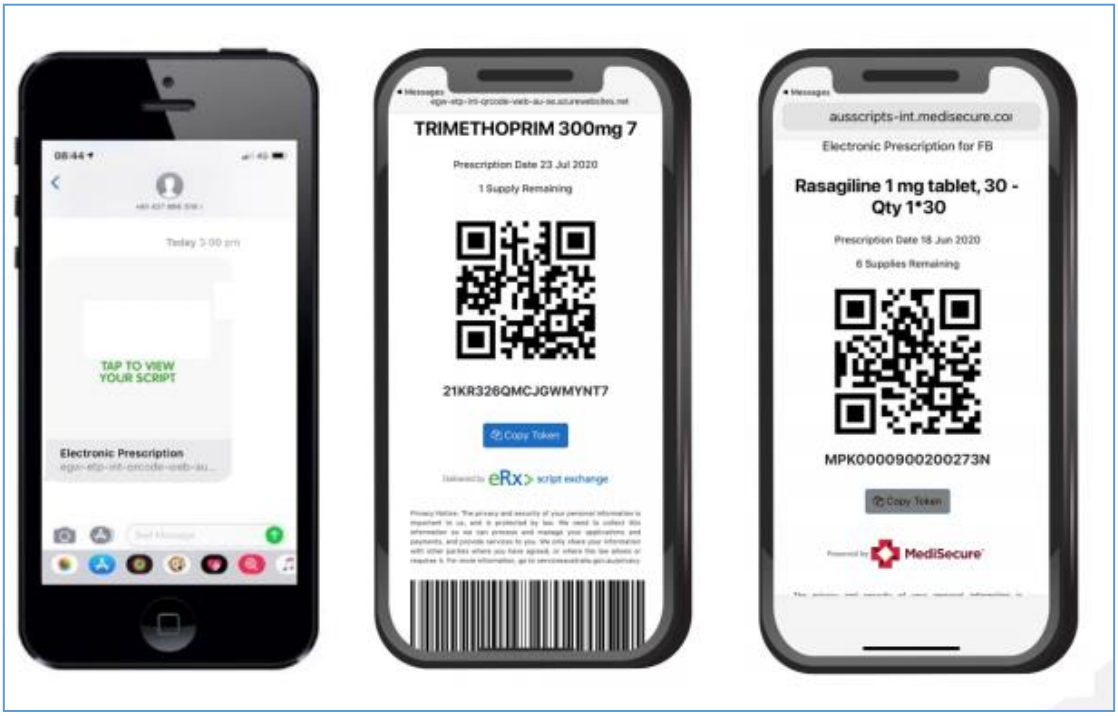



*Figure 5 - Examples of eRx and MediSecure barcodes on phones*

This system saw some great benefits with error rates dropping significantly by around 60% (18) and that service users were in control over their own prescriptions and took an active role in the management of their own medication. An overall time improvement was noticed as there was a reduction in duplication of prescriptions and the more efficient use of time by GPs and pharmacists (2).

There are some drawbacks that can be drawn from this project. With each prescription there comes a processing cost, each prescription whether it is MediSecure or eRx, there is an AU$ 0.15 cost (19). Although the service may be free this processing cost can amount to a lot for each pharmacy.

There are also security issues with this prescription system. Data while in transit depends entirely on the security and encryption from the platform it is sent and received from and there is no security for this information while it is in transit (18).

Another issue is that this prescription system is based on a centralized database. There are many issues with data stored in a centralized database as there is a reliance on a central server to perform all network operations and functionality. If this central server fails, there is a single point of failure for the whole system.

These issues will be taken into consideration when developing this project where security is at the forefront of the project and providing reassurance that the service user's information is secure.

While Australia's ePrescribing service provides clear improvements from both paper-based prescription systems and the service provided by Healthmail there are still core security issues and issues that can be addressed.

### 2.3.3 DMMS

DMMS is a decentralized blockchain ledger for the management of medication histories (7). This tool permits prescribers to create prescriptions for each service user and perform advanced searches of prescriptions used in the past. DMMS encrypts the prescriptions with the service user's public key where only the service user can decrypt their prescriptions using their own private key as seen in Figure 6. This would provide a good ePrescribing system where service users are in control of their own prescriptions only giving access to participants that should have access to it. This process cuts out the middleman. The results of this study show that this system can improve the security, trustworthiness, and privacy of medication history based on a centralized database or server. This project uses blockchain technology to improve and advance ePrescribing.

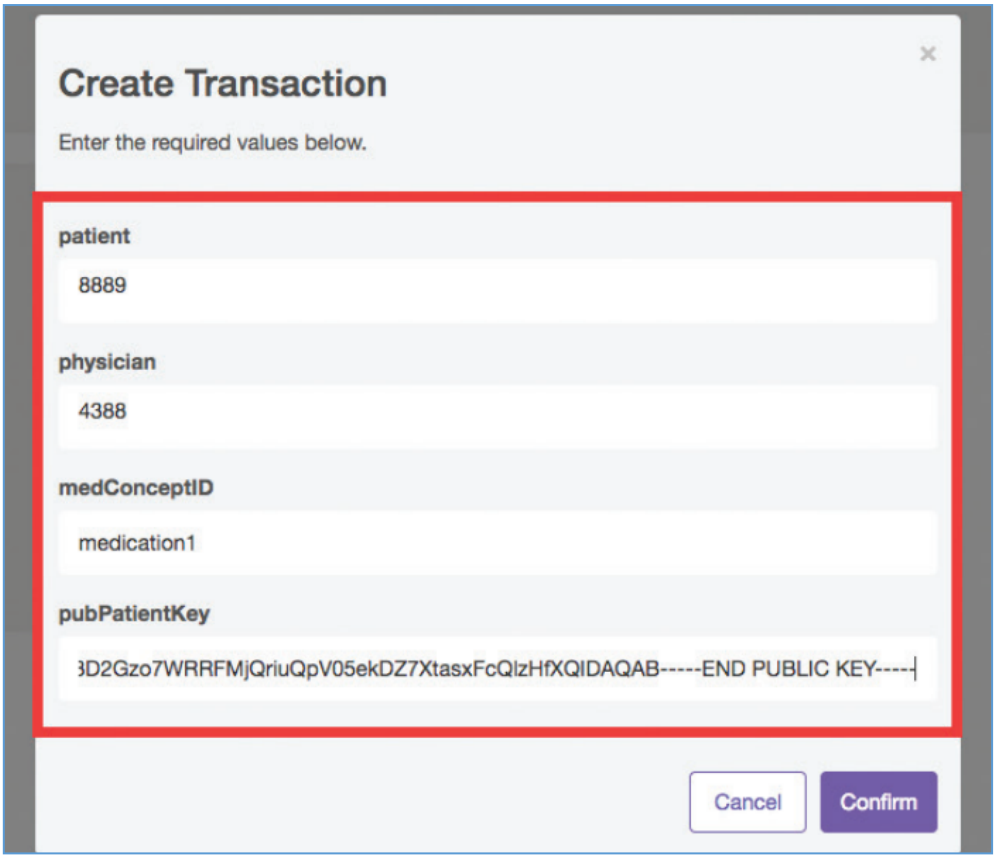



*Figure 6 - Transaction in the DMMS system- Transaction in the DMMS system*

Although this blockchain application represents an improvement to ePrescribing there are some slight improvements that could be made to this system. The blockchain could be incorporated into a shared system with the emergency department so that medication history and current medication could be accessed while the service user is in their care. This could save time in searching for medication history or waiting for access to be granted to the emergency department staff.

There is also the option to search a service user's information using the application. If a service user decided to clear all their information from the blockchain this would be impossible due to the immutability of the blockchain.

DMMS will be closely analysed and its issues focused on when developing the project due to the blockchain architecture it incorporates. While DMMS provides a clear example of blockchain's use for ePrescribing another example exists which is being used today.

### 2.3.4 X-Road (Estonia)

99% of Estonia's state services such as ePrescribing are online (20). Estonia has been at the forefront of digitizing government services since 2001 when the X-Road project began. X-Road is Estonia's solution to securely exchange sensitive or important data from different government organizations. It is a decentralized, distributed system that has utilized blockchain since 2008 (21).

X-Road's focus is data integrity and confidentiality of data. Data can only be accessed by those authorized to access it. X-Road uses the idea of an e-id which is an electronic identity which allows all Estonian citizens to access services on the X-Road.

For ePrescribing, the process starts with the GP or medical professional entering the service user's e-id which allows them to access medical records and medication history. A prescription is created and linked to the user's e-id which is then saved on the X-Road as seen in Figure 7. Information is saved to the X-Road using the cryptographic protocol to ensure data integrity.

The service user can then redeem this prescription at a pharmacy, the prescription can be redeemed once ownership is proved by using the service user's e-id.

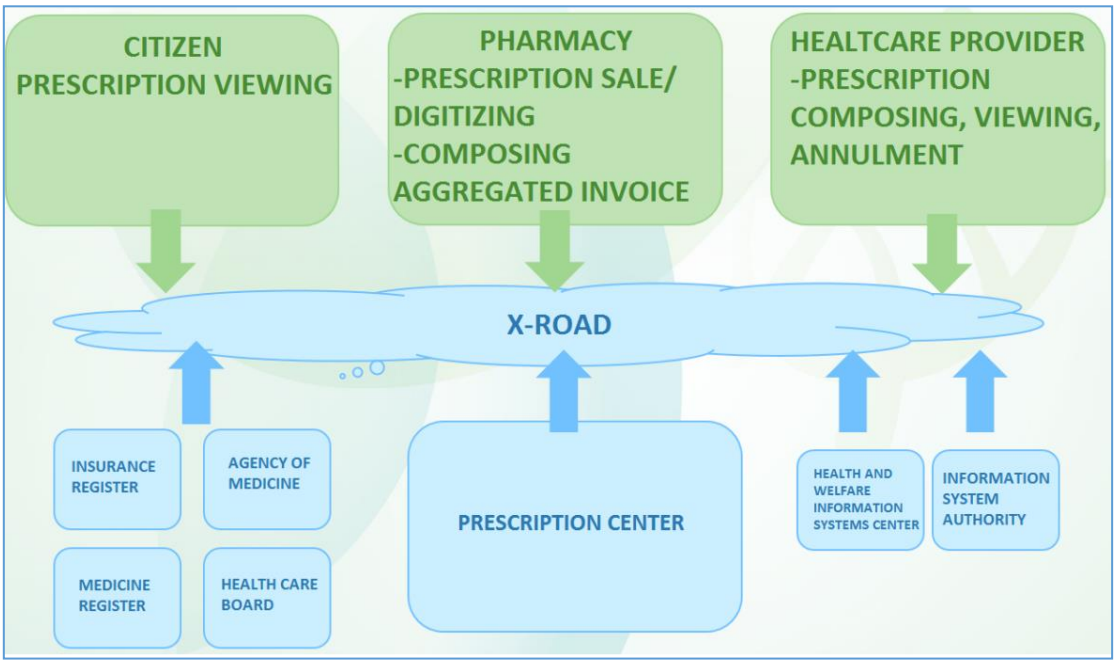



*Figure 7 - X-Road Organogram*

This prescription system and X-Road has seen huge successes. It is believed that nearly all prescriptions are now electronically processed (2). The error rate of prescriptions has drastically decreased with less than 0.01% of prescriptions in 2012 containing errors (22).

The cost of paper prescriptions has also drastically decreased as the majority of ePrescribing is now done electronically as seen in Figure 8. The costs of paper prescription forms have fallen 40 times since 2009 and are expected to fall even lower.

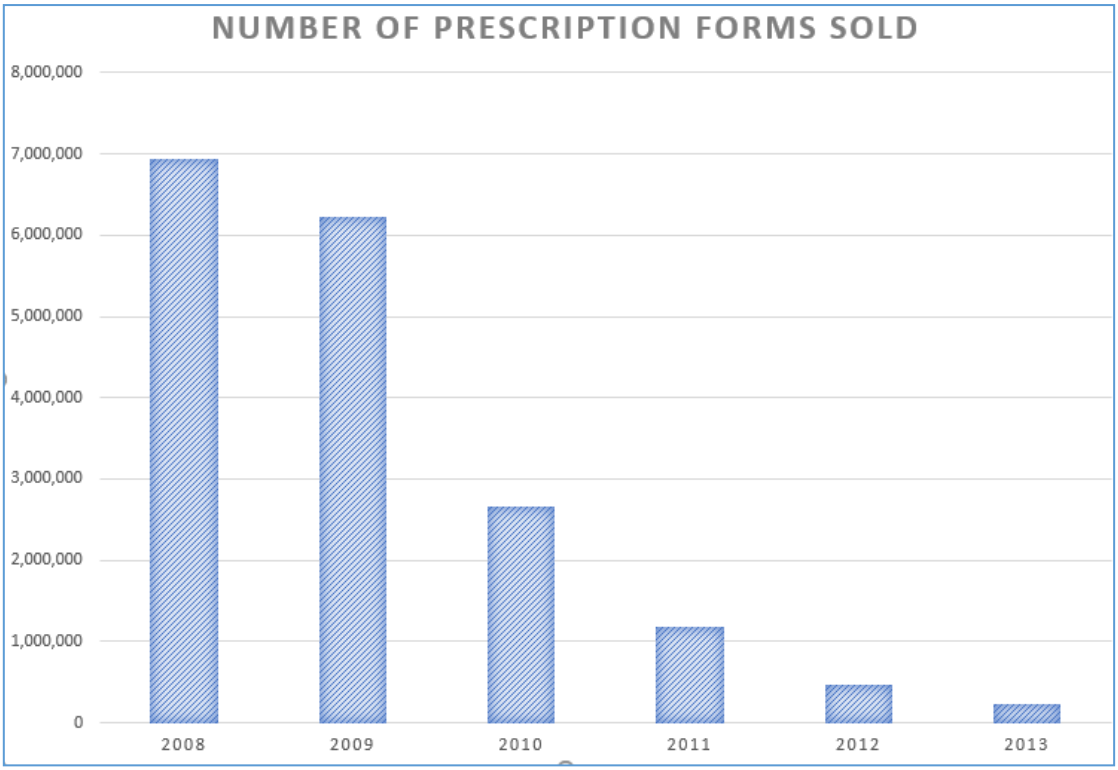



*Figure 8 - Number of Prescription Forms Sold*

Although X-Road and ePrescribing provide a positive development for ePrescribing there are some slight drawbacks that can still be improved or developed further.

Currently, ePrescribing prescriptions are saved in a centralized database even though it is based on the X-Road which is distributed. As mentioned previously centralized sources or single points of access can cause issues if the single source were to fail. Although the X-Road has proven itself to be reliable with a current up-time of over 1,100 consecutive days (23) this is still an aspect which can be improved upon.

### 2.3.5 Conclusion

In many of the systems explored, security was an issue that was present in most. Blockchain Technology does have similar functionality to these systems; however, the author was cognisant of these gaps and focused on the security of ePrescribing and securing service user's data.

## 2.4. Current Available Technologies

To explore if Blockchain Technology would attend to gaps in security (cf. Section 2.3.5), the author explored different technologies that are available. The author also provided the logo of each company as for some, the logo is more recognisable than the company name.

### 2.4.1. Python

Python was created by Guido van Rossum and released in 1991 and got its name from the comedy show on the BBC called 'Monty Python's Flying Circus' (24). Python is an object-oriented programming language. Python's appeal comes from its dynamic semantics and its high-level data structures which are built into the language (25).

It has quickly become one of the most popular programming languages to date with "The Importance of Being Earnest" or TIOBE for short, ranking it second with only the C language ahead (26). Python also has a huge variety of support for modules and packages which are easily accessible and installed using the PIP module which allows quick and easy installation. Another significant reason for Python's popularity is its simplistic syntax (cf. Figure 9). Python is more forgiving than most other languages and converts code to computer code itself. There is no need to worry about memory management (27).

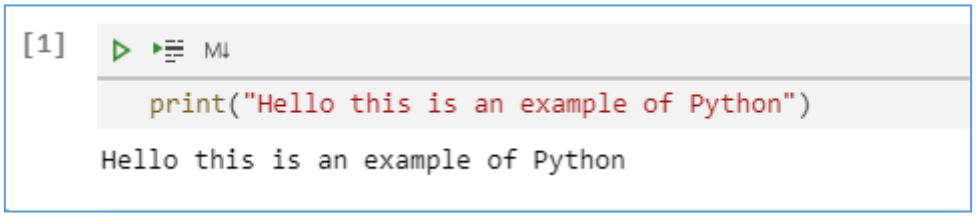



*Figure 9 - Python Simplistic Syntax*

Python integrates and is compatible with HTML and CSS and it also has benefits with both Hyperledger Fabric and INDY as both these blockchain frameworks can use python.

### 2.4.2. Java

Java was created by James Gosling and it was released in 1996. Its original name was "GreenTalk" and after that, it was called "Oak" and then after that due to an "Oak" already trademarked they renamed it "Java" (28). Java is an object-oriented programming language and operates from a high level. Java has proven itself as a robust language and its Java Runtime environment makes it compatible with almost all devices.

It is one of the most popular programming languages and is ranked third in the world on the TIOBE index (26). Java has huge support from powerful IDE's such as Eclipse and IntelliJ IDEA. Java also provides great documentation and allows developers to create documentation easily using JavaDocs.

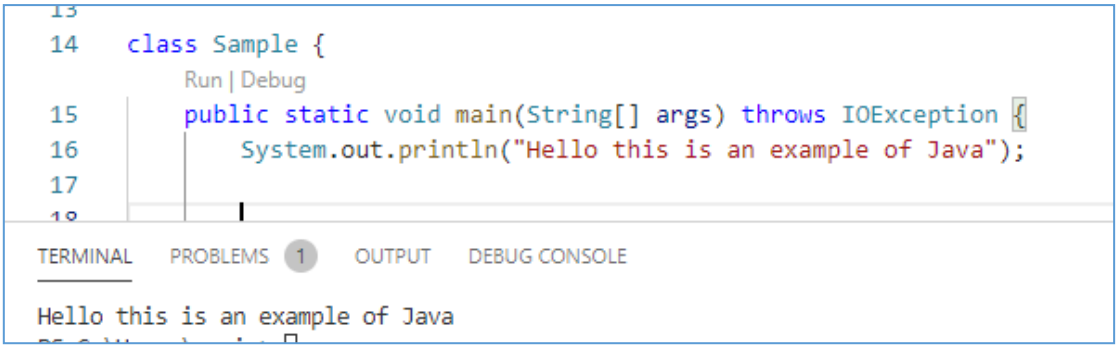



*Figure 10 - Example of java programming language*

Java is another language which interacts with web-based applications very well, Java is a strong backend to most web applications. Hyperledger also has example projects using Java which is very useful for this project.

### 2.4.3. MySQL

MySQL was created by a Swedish company called MySQL AB and was released in 1994(29). MySQL is a relational database management system that uses the query language SQL which stands for "Structured Query Language". MySQL presents itself as an easy-to-use and flexible database management system (DBMS) and MySQL is the current industry standard for most companies around the world (30). MySQL has proven itself to be one of the most popular DBMS to date with MySQL ranked 2nd in the DB-Engines ranking (31).

MySQL is a popular DBMS for web development which is a reason it was considered for this project. It is freely available and easy to understand. It integrates well with most object-oriented programming languages.

### 2.4.4. MongoDB

MongoDB was created by Dwight Merriman, Eliot Horowitz, and Kevin Ryan back in 2007(32). MongoDB is a database that stores information as documents or in other words JSON-like objects. It is a no-SQL database. Unlike MySQL which uses a relational database structure, MongoDB uses JSON objects to store data. MongoDB although much newer than MySQL is still relatively high on the DB-Engines index placing it at fifth on the list (31). Many reasons are credited for MongoDB's popularity. For one it is a no-SQL database which for some users makes database management much easier to understand especially if they are already familiar with JSON objects and their structure. MongoDB is also highly scalable and equally flexible (33). MongoDB would be another suitable choice for database storage for this project. Its use of JSON documents provides easy integration with sending and receiving information from different applications.

### 2.4.5. Hyperledger Fabric

Hyperledger fabric was part of the family of frameworks released in 2015 by Hyperledger. Hyperledger is a blockchain tool developed for businesses by the Linux Foundation and IBM. Fabric is a modular framework architecture used for developing blockchain applications. It is an open-source distributed blockchain solution that is highly modular and has a broad range of use cases for industry (34). It has multi-language support and a collection of sample projects to work from. It has a flexible approach to data privacy and has a permission architecture (34).

Hyperledger's dynamic framework and its enterprise standard for a distributed ledger is a useful feature. Its ability to keep data private while still operating a public blockchain is another benefit to this project.

### 2.4.6. Hyperledger Indy

Hyperledger Indy, like Hyperledger Fabric, was part of the family of frameworks released in 2015 by Hyperledger. Like Fabric, Indy is a distributed ledger that also provides tools, libraries, and reusable components for providing digital identities that are rooted in blockchain applications or on other distributed ledger platforms (35). This idea of digital identity gives the user full control over their own personal information which they disclose over the internet. Personal information is disclosed through peer to peer connection rather than stored on the ledger. This is the idea of self-sovereignty which also gives companies the confidence that a user's identity is true to who they are (36).

Hyperledger Indy has a straightforward API and provides strong security for user's data. Like Hyperledger fabric, Indy provides sample projects and guides in how to use Indy.

### 2.4.7. Docker

Docker was founded by Solomon Hykes back in 2010 and was originally called dotCloud (37). Docker is an open platform for developing applications that can have their own infrastructure separate from your own environment. These loosely isolated environments are called containers (38). These containers can run simultaneously with other containers and can be used to deploy applications quickly. Docker is very popular due to the current state of developing applications today which requires different architectures and environments for each application (39).

Docker is integrated with many of the Hyperledger products and is needed to deploy and develop the blockchain frameworks. Docker also will be useful in running and deploying separate nodes in the blockchain network.

### 2.4.8. Selenium

Selenium was founded in 2004 by Jason Huggins, Paul Gross, and Jie Tina Wang. It was called selenium due to a joke that selenium is an antidote for mercury which was a rival company at the time (40). Selenium is a free open-source automated testing framework used to validate different web applications; it can be used to test multiple programming languages. Selenium has four main components to its suite of software. These are Selenium IDE, Selenium RC, Selenium WebDriver, and Selenium Grid. Selenium WebDriver is also known as Selenium 2.0 allows you to operate across a variety of browsers using the JSON wire protocol. JSON write protocol allows communication between the client and the browser by creating an HTTP request in the form of JSON (41).

Selenium 2.0 is a good choice for automated testing for this project, it is at the forefront of web development testing and it can be used to test different programming languages.

### 2.4.9. JavaScript

JavaScript was first created by Brendan Eich in 1995, it has since then had many updates and changes to improve performance and standardization (42). JavaScript is a lightweight, object-oriented scripting language used in mainly web applications. JavaScript can be run both on the client-side and the server-side development. JavaScript also uses an easy to understand and easy to send object model called the document object model or DOM. JavaScript allows website functionality beyond the reaches of simple HTML and gives a more interactive experience with the user (cf. Figure 11). JavaScript has also been the base for many other languages such as Node.JS (42).

```
> console.log("This is an example of a print statement in JavaScript")
  This is an example of a print statement in JavaScript
```

*Figure 11 - JavaScript Print Statement*

JavaScript overall is a very popular web development tool and one of the most popular programming languages according to the TIOBE index which places JavaScript at number 7 (26).

JavaScript is a must for web development and will be integrated with this project along with the rest of the web development tools.

### 2.4.10. HTML

HTML was created in 1989 by Tim Berners-Lee and is deemed the start of the worldwide web.HTML stands for Hypertext Mark-up Language and is the standard mark-up for language for creating web pages. HTML describes the structure of a web page so that a browser can interpret it. HTML has been the backbone of web development for decades and continues to do so (44). HTML 5 is the latest version of HTML to date.

HTML is necessary for any web application and is included in this project.

### 2.4.11. CSS 

CSS was created in 1994 by Håkon Wium Lie, although it was not a new idea as the styling of websites was implemented within HTML, it was always the goal to separate document structure from its styles since the inception of HTML (45). CSS stands for cascading style sheets and allows the styling of web pages and describes how HTML elements are to be presented on the screen.

### 2.4.12. Maven 

Maven was created initially as part of the Apache Jakarta project in 2001 where the first import of prototype sources happened (46). Maven is a Yiddish word meaning “accumulator of knowledge” (47). Maven is a build automation tool which is a standard way to build projects and give a clear definition of what each project consists of such as the JARs included within the project. Maven’s primary objective is to make the build process easy, provide a uniform build system, provide quality project information, and encourage better development practices. Maven was one of the build automation tools researched for this project.

### 2.4.13. Gradle 

Gradle is a build automation tool which was first created in 2007(48). Gradle is similar to Maven in that it provides a standardized build for projects and give a clear definition of what the project contains. Gradle builds projects with a series of tasks which each build different components of the project. This gives Gradle an edge over maven as Gradle avoids unnecessary work by only running the tasks needed. Gradle also offers customisation of each task to better suit the developer’s needs.  Gradle was used in this project as Gradle integrates well with Hyperledger.

### 2.4.14. Tomcat 

Similar to Maven in Section 2.4.12, Tomcat was initially part of the Jakarta project and was first developed in 1999 (49). Tomcat became its own project in 2005. Tomcat is an open source implementation of a Java web server environment which allows different Java technologies such as Java servlet and Java web sockets to be deployed. It allows multiple Java web applications to be deployed using a WAR file and each can be monitored. Tomcat was an excellent choice for this project.

### 2.4.12 Conclusion

The technologies discussed in this section will be used to aid the development of this blockchain application. Their properties and the advantages these technologies bring will aid in the strengthening of this application.

## 2.5. Blockchain research

This section describes the other research regarding blockchain technology and its legality in today's modern age. The following chapter will discuss the architecture and structure of blockchain and what makes it appealing to healthcare developers. The laws surrounding blockchain and how blockchain interacts with information governance will also be discussed.

### 2.5.1. What is Blockchain?

Blockchain is a distributed immutable ledger of transactions which essentially means a decentralized database that uses cryptography to link each 'block' of data together (50) (cf. Figure 12). Each block of data or container of data contains both the data to be written to the ledger and the hash of the previous block. This linking of data or 'chain' of data gives the blockchain a clear history, as each block references the last block in the chain.

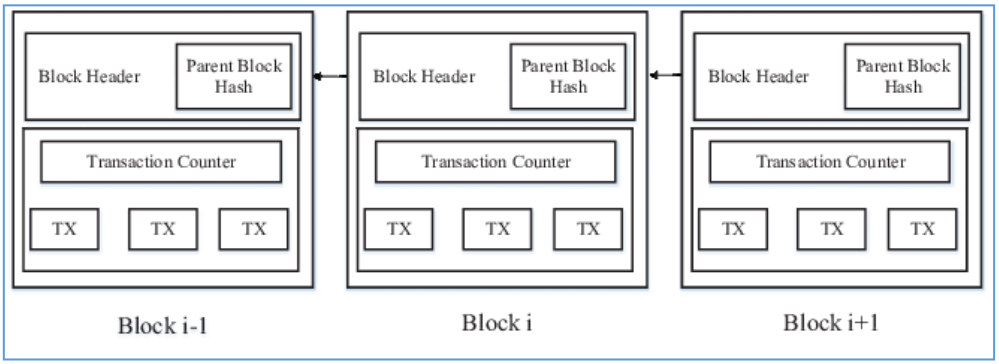



*Figure 12 - Blockchain Decentralised Database*

Distribution of blockchain refers to the fact there is no single entity that controls the validation of new entries on the ledger and instead, each node that is part of the network confirms transactions and adds it to the ledger (cf. Figure 13). Each node on the network must vote on the entries to be added to the network. Nodes with the longest history of the ledger and the most proof will usually be chosen. For each node, the ledger is updated, and once updated to the ledger the data cannot be changed or deleted which gives the data integrity. integrity. The network is said to be fault-tolerant, whereby as if one node goes down, there is still the rest of the remaining nodes will keep the network running and ledger up to date. Once the node comes back online the node's ledger will be updated (51). Overall blockchain provides data integrity without the use of a third party and a secure method of keeping this data safe.

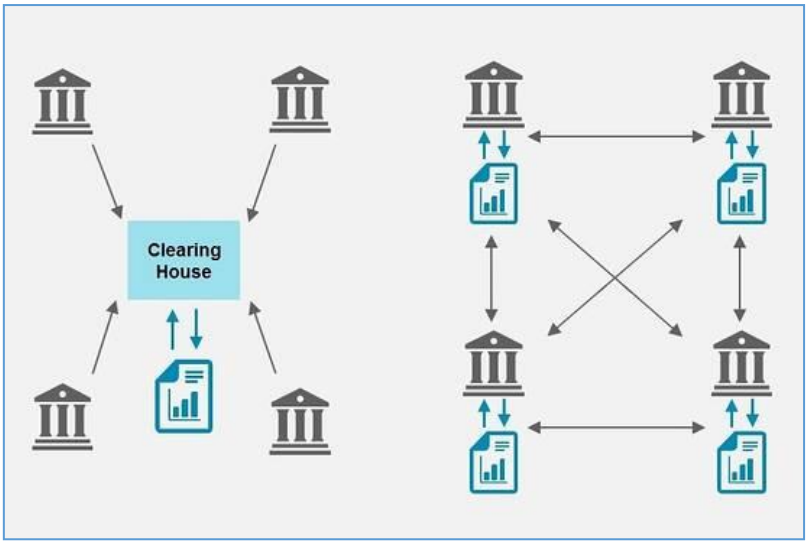



*Figure 13 - Blockchain Distribution*

Consensus algorithms are another critical part of blockchain networks. Consensus algorithms are how nodes specify new blocks to be added to the chain. There are many different consensus algorithms used by various different blockchain applications such as Proof-of-work (PoW), Proof-of-stake (PoS), and Practical byzantine fault tolerance (PBFT) but there are many more consensus algorithms available (51) as shown in Figure 14.

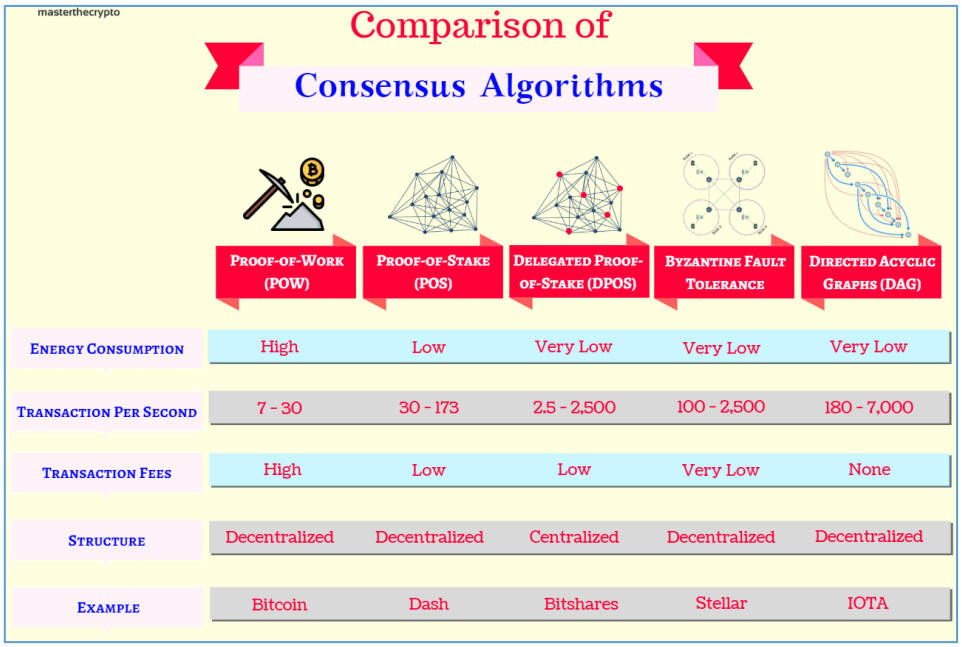



*Figure 14 - Comparison of Consensus Algorithms*

PoW is a process where each node of the network solves a cryptographic puzzle to calculate the hash of the next block and it can sometimes require a lot of computing power and energy to solve. Once a hash value is calculated, it is broadcast to the rest of the nodes on the network which will validate it. PoW operates in a way that hashes are difficult to calculate but very easy to validate (51). With each additional node that is added to the network, the network security increases.

PoS is a different method of consensus which takes a more energy-saving approach. PoS operates by using validator nodes on the network, which must prove ownership over the cryptocurrency the node owns. From this, a new block is credited to the validator node with either randomization or a combination of the size of the stake the validator holds (51).

PBFT is an algorithm that comes to an agreement between nodes in a system despite malicious or node failures, it operates in a majority rule where the malicious modes must not equal or exceed one-third of all nodes in the system (51). There are four rounds in this consensus mechanism:

- The client sends a request to the leader to invoke a service operation.
- The leader sends this request to the backup nodes.
- The nodes execute the request and send a reply to the client.
- The client waits for the replies from the other nodes with the same result.

This algorithm is similar to the PoW consensus algorithm as more nodes within the network provides the network with greater security (52).

Although blockchain technology has huge potential and advantages there are some issues with this technology. Once a blockchain systems architecture has been decided and put into motion, it is very difficult to change this architecture and scale it accordingly (51). Secondly, data stored on the blockchain is immutable which means it cannot be deleted or changed. Government regulations such as GDPR (8) require that any person can request that their personal information be removed. This is an issue with any information that is added to the ledger as once it is added it cannot be removed.

Blockchain comprises four main characteristics: Decentralization, Persistence, Anonymity, and Audibility. Decentralization refers to the use of consensus algorithms across a large network of nodes, there is no central body control and validating transactions. Persistency refers to the transactions that cannot be deleted or changed once recorded in the ledger. Anonymity refers to blockchain's namelessness of users who interact on the blockchain network. In lieu of personal details stored, each user has a secure identifier wallet address, although as mentioned this has some privacy risks associated with it. Audibility refers to the fact that each transaction that occurs on the blockchain can be traced to an address and each transaction can easily be verified. These characteristics are critical to blockchain's' success and popularity (50). Blockchain technology has seen uses in many different areas,

but mainly in areas where there is a need to prove ownership such as digital currencies or storing permanent information. A well-known example of blockchain is the digital currency Bitcoin (53).

### 2.5.2. Example of blockchain applications?

#### 2.5.2.1 Bitcoin

Bitcoin is a famous example of a blockchain application, it was credited with popularising blockchain technology and providing an appropriate use case for blockchain. It was created by Satoshi Nakamoto which is a pseudonym as the real creator is unknown (51). It was created in 2008 with the whitepaper "Bitcoin: A peer-to-peer electronic cash system" (53). This paper proposed a new form of payment that would not require a third party or financial institution to verify payments. Bitcoin uses a network of nodes, which run the bitcoin network to validate transactions and add them to the ledger. These transactions are grouped into blocks using time sequencing. New blocks are found by solving complex mathematical calculations. These calculations are difficult to solve but easy to prove, leading to the "proof of work" consensus algorithm (53).

Bitcoin started a new era of digital currencies with blockchain technology. Ethereum was one of these currencies.

#### 2.5.2.2. Ethereum

Ethereum is another digital currency backed by the technology of blockchain and is currently the second-largest cryptocurrency with a market cap of over €55 Billion (54). Ethereum was created by Vitalik Buterin in 2013, the project took a lot of inspiration from Bitcoin (55). Like Bitcoin, it is a decentralized blockchain application but Ethereum is more than just a digital currency but rather an ecosystem of blockchain development. Developers can use Ethereum to create their own decentralized applications or Dapps using the programming language Solidity on the Ethereum network (55). These Dapps are often referred to as smart contracts, are a set of 'IF' and 'ELSE' that execute when certain events are triggered (55). Once these smart contracts are deployed to the Ethereum network they cannot be controlled not even by the person who wrote the code (55). Ethereum currently uses two consensus algorithms, originally it used the proof of work algorithm to validate transactions and to execute smart contracts. Recently Ethereum 2.0 was announced which uses a new consensus algorithm which is proof of stake. Rather than nodes solving complex mathematical problems a node simply must have 32 Ether deposited in a smart contract, once complete that address becomes a validator node (56).

Ethereum brought a new wave of blockchain development to the ecosystem but both Ethereum and Bitcoin have a public ledger allowing anyone to view the transaction history of an address. Monero is another blockchain application that focuses on creating a private blockchain.

### 2.5.2.3. Monero

Monero is a cryptocurrency which focuses on privacy and decentralization of digital currency (57). Monero's white paper was first published in 2014 by the author Nicolas van Saberhagen which like Satoshi Nakamoto was a pseudonym. Monero's goal was to create a cryptocurrency which operated using blockchain technology but obscures the ledger so no outside parties can tell the source, amount, or destination of the payment sent or received (57). It uses ring signatures which is an anonymous cryptographic signature in which only the parties involved with the transaction can examine the full details of the transaction (58). As bitcoin operates on a public ledger, anyone can view the transaction history on any wallet address. Transactions can then be linked to wallet addresses with the amounts and frequency of transactions; it is entirely possible to uncover a person's identity by observing these patterns (59). This gave Monero an edge against bitcoin in terms of privacy, as Monero is a fungible currency where past transactions using the coin cannot be traced. Monero like Bitcoin uses a proof of work consensus algorithm to incentivize miners to validate transactions (57).

This fungibility property of Monero gave it its popularity, especially amongst users who were interested in buying things illegally on the dark web.

### 2.5.2.4. Hyperledger

Hyperledger is an open-source collection of frameworks, tools, and libraries for developing blockchain applications created by the Linux Foundation and IBM in 2015 (36). Unlike the previous examples of blockchain applications, Hyperledger's focus is to provide enterprise-grade blockchain technology across various industries so that blockchain solutions can be created for different technology fields (36). As Hyperledger is a collection of different blockchain frameworks and tools, each framework has a different blockchain architecture. While in each of these blockchain frameworks there are different consensus models, there is a generalized model of consensus which each framework adheres to which can be seen in figure 15.

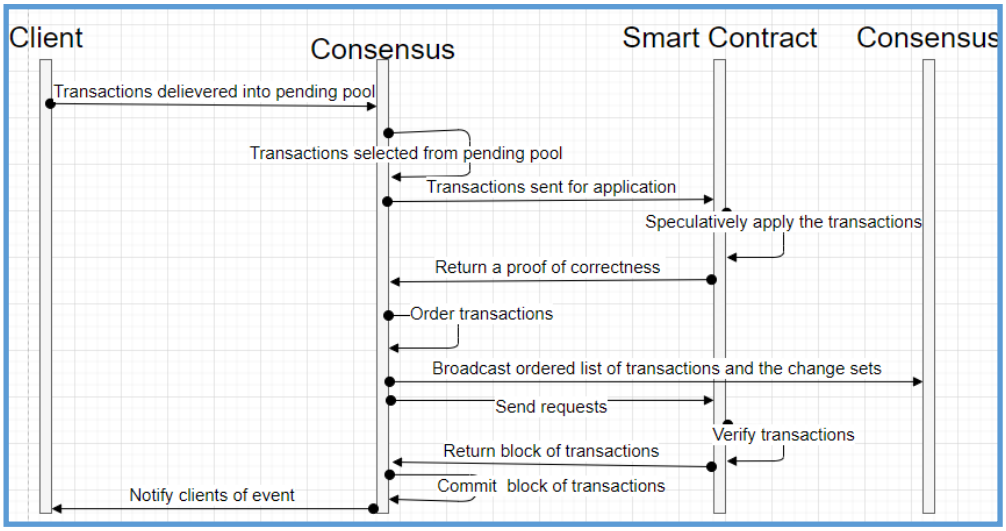



*Figure 15 - Generalized consensus model for Hyperledger*

Each of the frameworks provides a high level of security and an easy to use Application Programming Interface (API) (60), they do not have any cryptocurrencies as Hyperledger's focus is on different applications of blockchain technology. Hyperledger also provides a private blockchain for development, like Monero, except access to the blockchain is restricted and limited to only those who are authorized to view the blockchain.

Private blockchain's have some advantages over public blockchain's as they fulfil data confidentiality requirements as data is only accessed by those who are authorized to (61).

This private blockchain network along with tools and materials to develop blockchain applications have made Hyperledger a powerful and popular platform. 30 companies in the Forbes Blockchain 50 were using a Hyperledger product to develop their applications (62).

Blockchain has many applications especially in finance, but these fundamental attributes of decentralization, persistence, anonymity, and audibility are excluded to these fields, blockchain has an emerging interest in how it can improve health care and more specifically the security of health information.

### 2.5.3. How can blockchain be used in Healthcare?

Blockchain technology has many potential uses in the healthcare sector as it could help enhance the infrastructure in the sector. Areas such as medical device tracking, clinical trials, pharmaceutical tracing, and health insurance (63). Blockchain can solve a variety of healthcare's problems with its four main characteristics of decentralization, persistence, anonymity, and audibility (51). These key characteristics of blockchain can give data integrity, help prevent fraud, and create a 'trustless' system without the need for a third party to monitor it (63).

Patient records are another area that could see huge improvements from blockchain technology (63). Blockchain has the power to put sensitive data in the hands of the patient, allowing patients the control of their data while maintaining a clear immutable record of medical history for patients. Health records of patients also contain significant amounts of duplicated information, blockchain can make this information more dynamic by creating profiles that could be linked to different types of documents. One electronic identification that would be verifiable and trusted could be created to be linked to each medical document rather than repetition and replication that is currently in healthcare systems today. A challenge to this would be to replace all current documents which would take a significant amount of resources and time.

Drug tracking is another area which could see advances in development using blockchain technology. The immutability property of a blockchain ledger provides a clear history of transactions that occur, this attribute would be especially important for tracking the supply chain of pharmaceutical drugs (63). This would allow healthcare providers to meet current standards for pharmaceutical supply security such as the European Union directive 2001/83/EC for the security and validation of medical products (64). This use of blockchain to verify pharmaceutical drugs can prevent fraud and theft of pharmaceuticals.

These are just two of the examples of benefits blockchain can bring to the healthcare sector with many more emerging as blockchain support and understanding develops. Blockchain's fundamentals of decentralization, persistence, anonymity, and audibility are attractive characteristics for technology in the healthcare industry where data integrity is key.

### 2.5.4. Legislation and regulations

The author developed a blockchain application for an ePrescribing system which required relevant research to be conducted into the legislation and regulations surrounding ePrescribing in Ireland. Currently, Ireland still has strict legislation surrounding prescriptions and they must be written or typed onto a physical prescription paper (65). Temporary amendments were made to this regulation to allow the transfer of prescriptions through an email service called Healthmail (4) although this was a step in the right direction, prescriptions were not created electronically but rather written out and then scanned to be sent electronically.

GDPR (8) is another piece of legislation that is central to information governance. GDPR is a privacy and security law drafted and passed by the European Union in 2018 (8). It provides strict regulations with regards to data protection and how data is managed and used by companies and applies to any EU citizen or resident data that is processed. GDPR can be broken down into 7 core principles as below:

1. Lawfulness, fairness and transparency — Processing must be lawful, fair, and transparent to the data subject.
2. Purpose limitation — You must process data for the legitimate purposes specified explicitly to the data subject when you collected it.
3. Data minimization — You should collect and process only as much data as absolutely necessary for the purposes specified.
4. Accuracy — You must keep personal data accurate and up to date.
5. Storage limitation — You may only store personally identifying data for as long as necessary for the specified purpose.
6. Integrity and confidentiality — Processing must be done in such a way as to ensure appropriate security, integrity, and confidentiality (e.g. by using encryption).
7. Accountability — The data controller is responsible for being able to demonstrate GDPR compliance with all of these principles (66).

GDPR also gives individuals more control over the data they give to organizations and make individuals aware of the data they are giving. There are also rights for each data subject or person who uses the internet, these rights are:

1. The right to be informed.
2. The right of access.
3. The right to rectification.
4. The right to erasure.
5. The right to restrict processing.
6. The right to data portability.
7. The right to object.

8. Rights in relation to automated decision making and profiling (66).

If a company breaches GDPR legislation, the company will face a costly fine. These fines are determined by the severity of the infringement against GDPR with less severe infringements resulting in a fine of up to €10 million or 2% of the firm's worldwide annual revenue while more severe infringements could result in a fine of up to €20 million or 4% of the firm's worldwide annual revenue (67).

Blockchain at a first glance appears to follow GDPR as one of the key characteristics of blockchain is anonymity (55) and the scope of GDPR only applies to personal data. Blockchain uses primarily wallet addresses and encryption keys to send and receive information on the blockchain. However public blockchain's in which the ledger can be viewed publicly, can be analysed, and can deduce information about users of the blockchain (63). Chain analysis can be used to get personal information from transactions linked with a private key and in some cases, an entity can identify the person behind the private key. Public blockchain applications are therefore not anonymous but rather pseudonymous (68). Since pseudo-anonymity still has the possibility of revealing personal information it would still fall under the scope of GDPR.

Another aspect of blockchain technology that could cause issues with GDPR is that most data or information is not processed by one single entity but rather a collection of nodes on a network each with their own copy of the data. While in private blockchain's where one company can be associated with the blockchain, public blockchain's do not have a single entity controlling the processing of data. This grey area in terms of data controller could lead to a huge downfall in blockchain technology adoption (68).

Finally, the last aspect focuses on the 'right to erasure' (8). Blockchain operates on a system where entries into a ledger cannot be deleted, this gives the data integrity on the blockchain, but this causes issues with the right to be forgotten. If any personal data was stored on the blockchain it would be there permanently and would be in violation of this right of erasure (68). GDPR has some circumstances where the right to erasure does not apply, one of these circumstances is if the data processing is necessary for public health purposes in the public interest and if the processing is necessary for the purposes of preventative or occupational medicine or for the management of health systems or services (8).

In relation to ePrescribing standards, HIQA published the standards of information required for ePrescribing applications (1).
This standard was composed of the minimum dataset for a dispensing note or in this case ePrescription and defined as a Health Level 7 clinical document. This standard does not attempt to cover all information which should be recorded at a pharmacy as this is a much broader dataset but rather the minimum information needed (1).

There are 3 main parts of a prescription: Details of the patient, Details of the prescribing health professional, and the prescribed products (69). These datasets were closely examined when designing this project as identified in the following Table 1, 2 and 3.

| Name | Definition | Optionality | Usage |
|---|---|---|---|
| 1.1 Title | Coded value that contains the title relevant to the subject of care. | Optional | To be selected from a predefined list. |
| 1.2 Forename | A patient's first name or given name(s) as per their birth certificate. | Mandatory | A patient's first name or given name (s) as per their birth certificate. |
| 1.3 Surname | The second part of a patient's name which denotes their family or marital name. | Mandatory | The second part of a patient's name which denotes their family or marital name. |
| 1.4 Address | The location to be used to contact or correspond with the patient. This would normally be the patient's usual home address. | Mandatory | The particulars of the place where the patient lives. |
| 1.5 Date of Birth | Date of birth indicating the day, month, and year when the patient was born. | Mandatory | The date of birth should be supplied in dd/mm/yyyy format. |
| 1.6 Gender | Gender identity is a person's sense of identification with either the male or female sex, as manifested in appearance, behaviour and other aspects of a person's life. | Mandatory | Gender identity is a person's sense of identification with either the male or female sex, as manifested in appearance, behaviour and other aspects of a person's life. |
| 1.7 Health identifier | A number or code assigned to an individual to uniquely identify the individual within an organisation. | Mandatory | Both the code and the code type the code relates to should be provided e.g. 0987654321 Healthcare Record Number (HcRN). Other identifiers which may be carried in this field include the General Medical Scheme, Drug Payment Scheme, Long term illness scheme and Hardship scheme identifier. |

*Table 1 - Patient Details*

| Name | Definition | Optionality | Usage |
|---|---|---|---|
| Author ID number | The identifier number of the health practitioner who is responsible for dispense note. | Mandatory | The number or code assigned to the professional by its regulatory body or the health service's provider's identifier when it is implemented. This could be the superintendent pharmacist. The author is the individual that has logged in at the terminal at the time of record generation. |
| Author title | Coded value that contains the title relevant to the author of the document. | Optional | To be selected from a predefined list. |
| Author forename | The author's first or given name(s) as per their birth certificate. | Mandatory | Where the author is registered with a regulatory body, the forename should be the forename registered with the regulatory body. |
| Author surname | The second part of the author's name which denotes their family or marital name. | Mandatory | Where the author is registered with a regulatory body the surname should be the surname registered with the regulatory body. |
| Author profession | Coded element that specifies the author's particular profession. | Mandatory | This value can be selected from a predefined list. |
| Author telephone number | The author's telephone number. | Mandatory | The phone number to contact the author. |
| Author's email address | The author's email address. | Mandatory | The secure email address to contact the author. |
| Author's address | The particulars of the place used to correspond with the author i.e. the name and address and of the of the pharmacy premises where the medication was dispensed | Mandatory | The particulars of the place used to correspond with the author. |

*Table 2 - Details of the prescribing health professional*

| Name | Definition | Optionality | Usage |
|---|---|---|---|
| 1.1 Date of creation of dispensing note | The date (and optionally time) when a pharmacist created a dispense note of item(s) for the patient. | Mandatory | Date field which indicates when the prescription was dispensed. |
| 1.2 Medicinal Product | The name of the medicinal product or package. This should be sufficient to identify the medicinal product dispensed. It may be a trade name or a generic name. | Mandatory | A coded textual description associated with the medicinal product. |
| 1.3 Medicinal Product ID | A unique identification number associated with the medicinal product referred to in 1.2. | Optional | This will cater for a product-id for a national product catalogue. |
| 1.4 Medicinal product package | Size and or type of package prescribed. | Optional | When prescribing occurs at a package level, this field is used to describe the size and type of the package to dispense. |
| 1.5 Number of packages | Number of complete packages required to fulfil the prescription. | Optional | When prescribing occurs at a package level, this field is used to describe the number of the package(s) to dispense. |
| 1.6 Dose form (strength) | Content of the active ingredient expressed quantitatively per dosage unit, per unit of volume or per unit of weight, according to the pharmaceutical dose form. | Conditional | This field consists of a size value and unit, a combination of both defines the strength, for example 250mg or 1g. If 1.4 and 1.5 are populated, then this field does not need to be populated. |
| 1.7 Dose form (type) | A description of the dose type. | Conditional | This field describes the dose type, such as tablet or vial. If 1.4 and 1.5 are populated, then this field does not need to be populated. |
| 1.8 Total number of dose instances | Total number of instances of the medicinal product supplied to the patient by the pharmacist | Conditional | This field is used to describe the number of the unit(s) to dispense. If 1.4 and 1.5 are populated, then this field does not need to be populated. |
| 1.9 Instruction | Instructions for the medication. | Mandatory | A textual description associated with instructions to the subject of care. |

| | | | |
|---|---|---|---|
| 1.10 Comments | Any additional information that may be needed to ensure the continuity of supply, proper use, or appropriate medication management. | Optional | A textual description associated with additional information. |

*Table 3 - Details of Prescribed Products*

### 2.5.5. Conclusion

This section reviewed relevant research into blockchain technology and the legal implications surrounding it. It explained the architecture of blockchain and how this architecture provides many security benefits that can be utilized for use in a variety of different industries. This section also explored in greater detail the use of blockchain in medical applications and the benefits of blockchain technology in those applications. This section also investigated the laws and regulations surrounding blockchain and personal data, with regards to both national and international laws such as GDPR. It also considered the current standards and requirements of prescriptions in Ireland. This research was used to develop this blockchain application.

## 2.6. Existing Final Year Projects

Another direction in research for blockchain applications was to explore the blockchain projects from previous students and two final year projects which were closely related to this project were reviewed.

### 2.6.1. Identiphone - Final Year Project

The first final year project which was explored is the project produced by Aaron Byrne entitled “Identiphone” (70). This project investigated the use of blockchain technology to prove ownership of mobile phones. This was a preventive measure to help mitigate the risk of phone theft and create a solid foundation for phone ownership.

The system was composed of three tiers and was designed as a web application. The use of TypeScript, a form of JavaScript was used as the front end of the application. The middle tier of this application was a REST server which handled the requests made by the user from the front end while the backend consisted of a combination of Hyperledger Fabric, a blockchain framework, and MongoDB, a NoSQL database. Identiphone’s approach to a blockchain application is a similar approach to how this current project will be designed. The blockchain framework that was used, Hyperledger fabric which has some definite use cases for blockchain applications which are not cryptocurrencies.

There were some issues with setting up the blockchain network to include more than one node. This meant that the security of this program could not be fully tested. This issue will be taken into consideration when developing this project.

### 2.6.2. Pharmap - Final Year Project

The second final project is the project produced by Nicola Mahon entitled" A Blockchain Risk Mitigation Solution for the Pharmaceutical Supply Chain" or abbreviated as Pharmap (71). This project proposed the use of blockchain to map the flow of pharmaceutical products throughout their distribution life cycle to provide traceability on the market.

This project, like the previous project, was developed as a 3-tiered web application. It used HTML and CSS as its frontend while using a rest server allowing interaction between itself and the frontend. The backend was a combination of MongoDB and Hyperledger Fabric. Again, this current project will be developed in a very similar way to this project's architecture with the use of a 3-tier structure.

The research needed to complete this FYP was unmanageable and therefore this project was not fully completed. This issue was taken into consideration when researching the project, research was limited to the fields I needed to complete this project.

## 2.7. Conclusions

In this chapter, relevant research on blockchain and ePrescribing was explored. First, the current ePrescribing systems available in both Ireland and on an international level were explored, which gave insight into the benefits and drawbacks of all ePrescribing systems. These advantages and disadvantages helped guide the design of this project and the potential challenges that may be encountered.

Technologies relevant to the development of this project were examined and researched, blockchain frameworks were especially analysed to ensure they had the appropriate capabilities needed for the success of this project.

Further research was conducted to investigate blockchain and blockchain's place in healthcare. Blockchain's architecture and different types of blockchain applications were researched to give a better understanding of how blockchain works. Research into current legislation surrounding prescriptions and personal data, such as GDPR was also explored which was necessary to understand the full aspects of blockchain's legality and place in data protection. This research will prove useful in developing an application compliant with current standards and legislation.

Finally, two final year projects were discussed, to give insight into how their blockchain applications were developed and the issues they encountered to help avoid the same issues happening again with this project. Overall, this research was necessary to fully understand the scope and the challenges of blockchain and the guidelines for developing blockchain applications.

# 3. System Design

## 3.1 Introduction

Following on from the research reviewed in section 2.7, the design of this ePrescribing blockchain system is discussed in chapter 3. The author will outline the methodology used in this project and other methodologies that could have been used will be outlined. This is followed by a discussion at each level of the architecture of this project. The levels of architecture which are discussed in this section are as follows:

- Front-End, which describes the user experience and what the user is presented with this application, this will be detailed using visual prototypes, wireframes that depict the user interface and use case diagrams that detail the interaction between the users and the system.
- Middle-Tier, which describes the different applications in this project, how they interact with each other and how they are used.
- Back-End, which describes the structure of database and blockchain technology, using Entity Relationship Diagrams (ERD) and Unified Modelling Language diagrams (UML).

## 3.2. Software Methodology

There are many different software methodologies that were examined for this project, this section describes each of the methodologies which were examined and then describes in detail the methodology chosen for this project.

### 3.2.1. Waterfall methodology

The waterfall methodology is a known methodology that has been around since 1970 and was created by Winston Royce (72). The waterfall is broken down into 7 main stages which are seen in Figure 16.

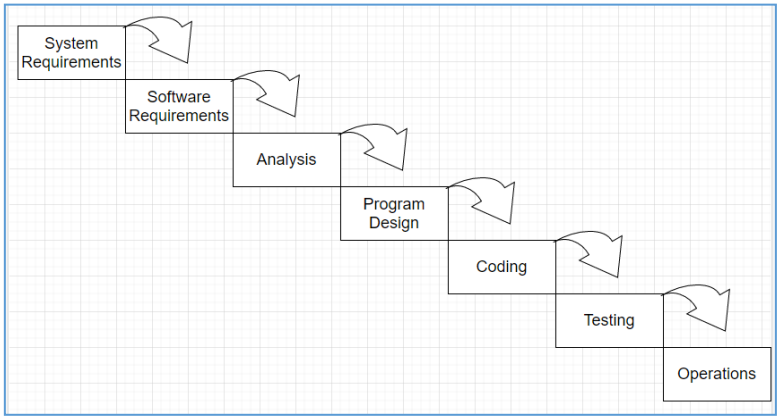



*Figure 16 - Waterfall methodology stages*

When one stage is completed the next stage is commenced and so on, hence the waterfall methodology has proven to be very simple to understand. The waterfall is normally used when the requirements of the project are clearly defined and there is low risk of the requirements changing (73). The waterfall methodology is widely considered to be rigid because of its linear approach to software development (73). Software development today requires a more flexible approach to software development due to ever changing system requirements from clients. Waterfall methodologies structure did not suit the development of this project as this project does not take a linear development approach.

### 3.2.2. Agile methodology/ Scrum

Agile 'Software development' Manifesto was created in 2001 by a group of developers (74). Agile focuses on an iterative approach to software development, to help fit the requirements of a customer throughout the development. Unlike the waterfall approach, Agile focuses on small increments of work and evaluates continuously throughout the development process. Agile methodology is not strictly a precise methodology but more a set of principles to follow. These principles are found within the Agile Manifesto which can be seen in figure 17.

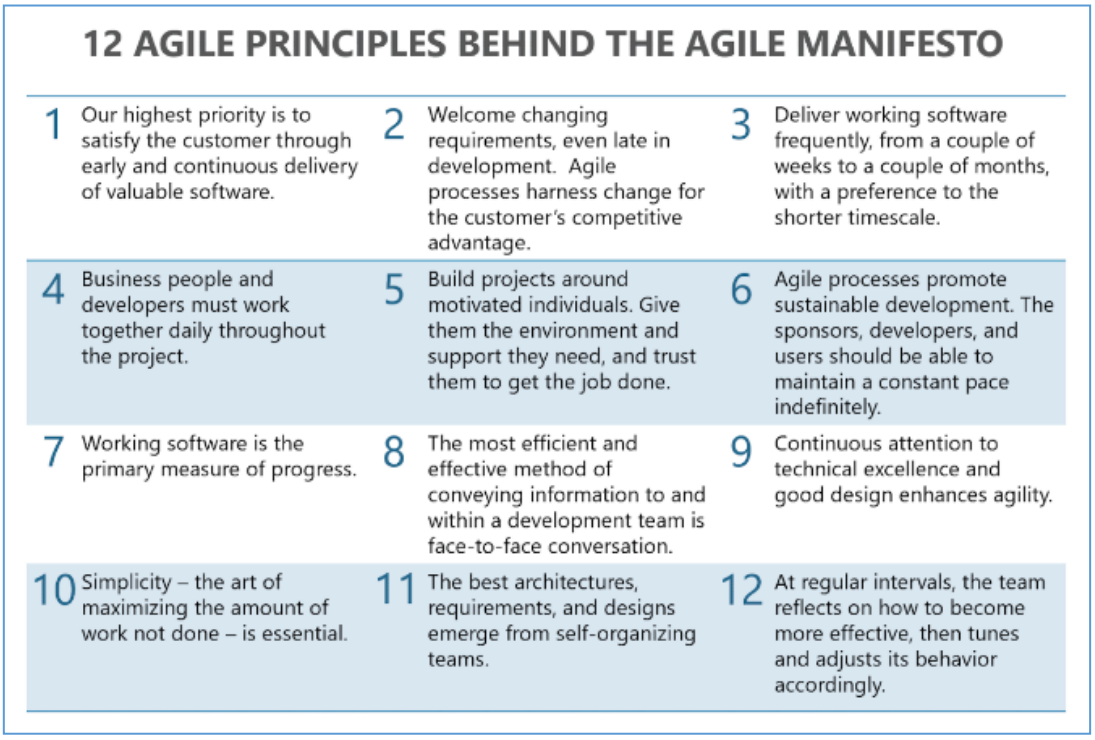



*Figure 17 - Agile manifesto principles*

Overall Agile focuses on a flexible and adaptable approach to software development. From this manifesto, many different methodologies were created, such as Extreme Programming, Feature Driven Development and Scrum (75).

Scrum was created by Ken Schwaber and Jeff Sutherland, who were also creators of Agile. Scrum is a lightweight methodology which is easy to understand and is based on the key principles of Agile (cf. Figure 18). Scrum has 3 main phases which are:

1. Pregame
    a. Planning: Definition of new release, based on the current backlog, or in other words requirements needed. The phase consists of analysis and conceptualization of the system.
    b. Architecture: Design how the backlog will be implemented into this system.
2. Game
    a. Development Sprints: Development of a new release, these releases are a combination of requirements found in the backlog. Programming, designing and constant communication between the team members are just some of activities found in a sprint cycle. Each sprint lasts for about 2-4 weeks.
3. Postgame
    a. Closure: Preparation for next release, including final documentation and pre-release stage testing (76).

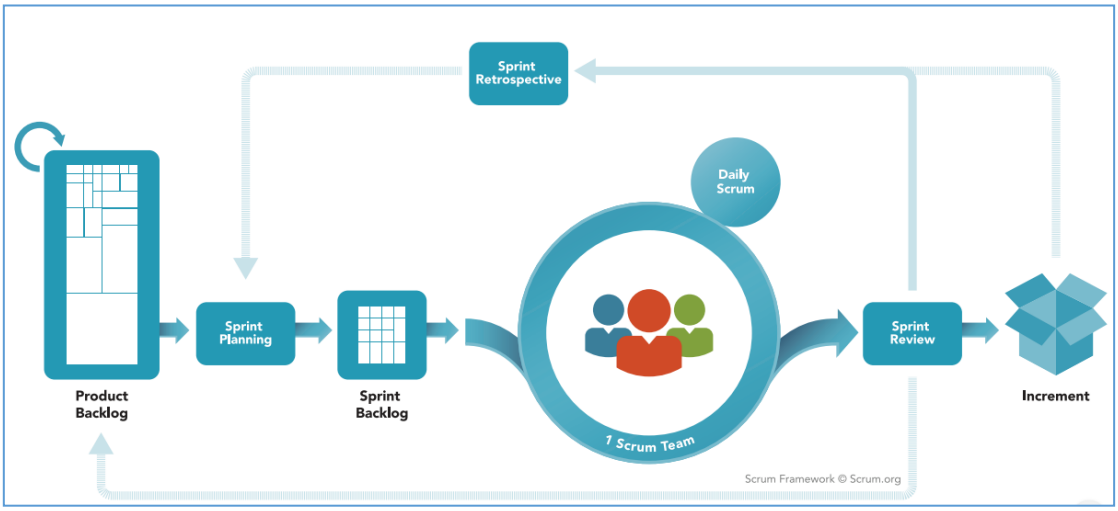



*Figure 18 - Scrum framework*

Scrum is usually suitable for big projects and teams of usually between 5-9 people, but this framework can be adapted to suit the needs of any project and any sized team with its flexible structure. This project uses the Scrum framework to structure the development process in an organized way. The basic structure of Scrum provides a solid foundation to maximise productivity and plan efficiently with the use of development sprints. This project uses the built-in project management system in GitHub to plan out the tasks for each sprint on the Scrum board. The Scrum board is separated into 5 sections:

- Backlog: Which contains all tasks which need to be completed, with a set priority assigned to each task.
- Research & Investigation: Which contains tasks which are currently undergoing more in-depth research.
- In progress: Which contains tasks which are currently being developed.

- Testing: Which contains tasks which are currently finished development and now being tested for errors and flaws.
- Done: Tasks which have been completed.

An example of some tasks that are currently in the backlog can be seen in figure 19

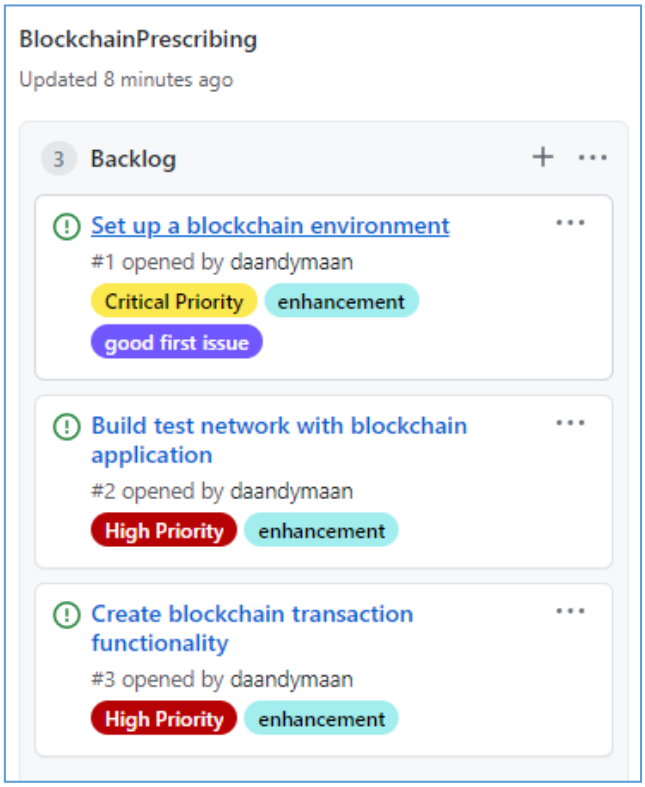



*Figure 19 - Scrum board backlog*

### 3.2.3. Test-Driven development

Test-driven development is a programming methodology which focuses on coding, testing and design. It was created by Kent Beck and used in XP, which is eXtreme Programming another Agile based Methodology (77). It can be broken down into a set of rules which are easy to navigate. These rules are:

1. Write a single test describing an aspect of the program (Testing).
2. Run the test to which the program will fail (Testing).
3. Write code to pass the test but only write the code that is necessary (Coding).
4. Refactor the code so that it appears simplistic (Design).
5. Repeat the process with a new test and make sure all previous tests still pass (77).

Test driven development has many advantages, such as the obvious benefit of constant functional code with a focus on keeping the code working to other benefits such as standardized code which is easy to follow and understand (78).

Some aspects of Test-Driven development are used in this project such as standardized code and refactoring code to provide better readability. Testing is another important aspect of this project with a big focus on security testing and unit testing for all components.

## 3.3. System Architecture

This section illustrates and describes the architecture, the requirements identified and the flow of the application.

### 3.3.1 Architecture

The architecture of this system is a three-tiered architecture which consists of a presentation layer, application or logic layer and finally a data layer (cf. Figure 20). This architecture is the basis for many web-based applications.

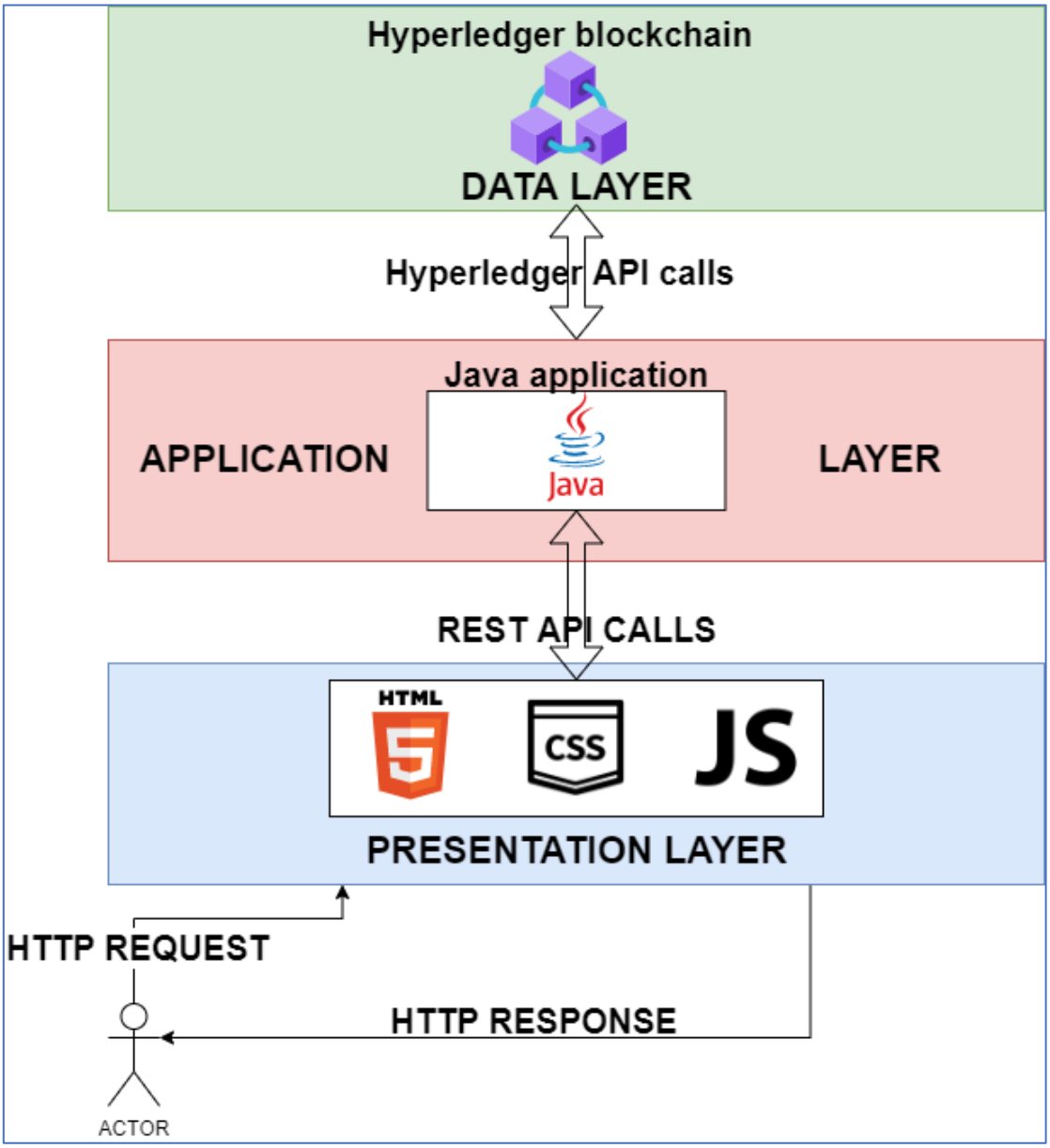



*Figure 20 - Overview of the architecture*

The presentation layer is a combination of HTML, CSS and JavaScript to create a functional web application. The user can interact with the web application using HTTP requests and in return the web application provides HTTP Responses. These three technologies are used as they are easy tools to use but can be used to create a simplistic styled but clear web interface.

The application or logic layer is composed of a Java application which acts as an intermediary between web application and the blockchain infrastructure in the data layer of this application. The application handles requests made by the user by using API endpoints. API's are "Application Programming Interface" and REST is "Representational State Transfer" (79). It allows communication between two systems using HTTP requests and HTTP responses. This allows communication between the presentation layer and the application layer. The application also communicates with the data layer, which contains the Hyperledger blockchain. The logic layer uses another set of API calls to connect to the Hyperledger blockchain. The Java application handles the results of these requests and returns a response to the presentation layer which handles it accordingly.

The data layer is composed of a blockchain network which are used to retrieve and save data requested by the application layer. The blockchain, which is from Hyperledger, is used for storing sensitive data and data that is permanently stored.

Hyperledger's blockchain also has an underlying architecture, as blockchain is distributed ledger technology which means it is not centralized and instead spread across multiple different nodes. Hyperledger offers a range of SDK which allow a simple connection between an application and the blockchain it provides. In Figure 21, a visual representation of the Hyperledger network is provided. Each groups of nodes which run the Hyperledger blockchain are broken down into organisations, these nodes validate transactions on the blockchain using a consensus algorithm which is what the order service represents.

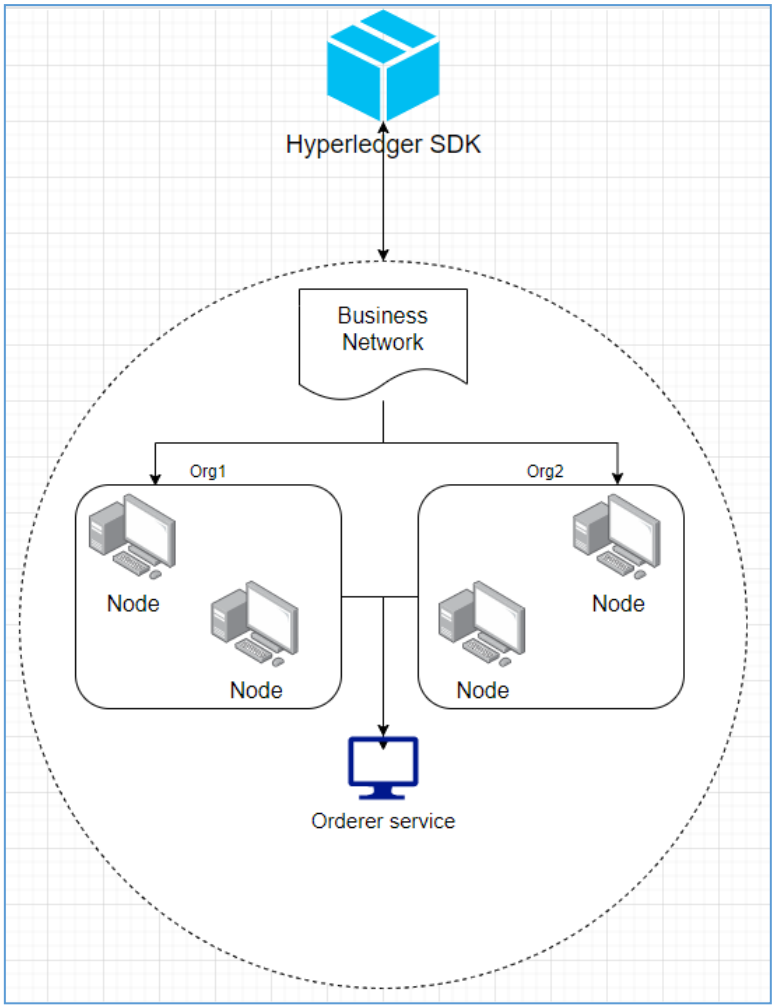



*Figure 21 - Overview of the Hyperledger architecture*

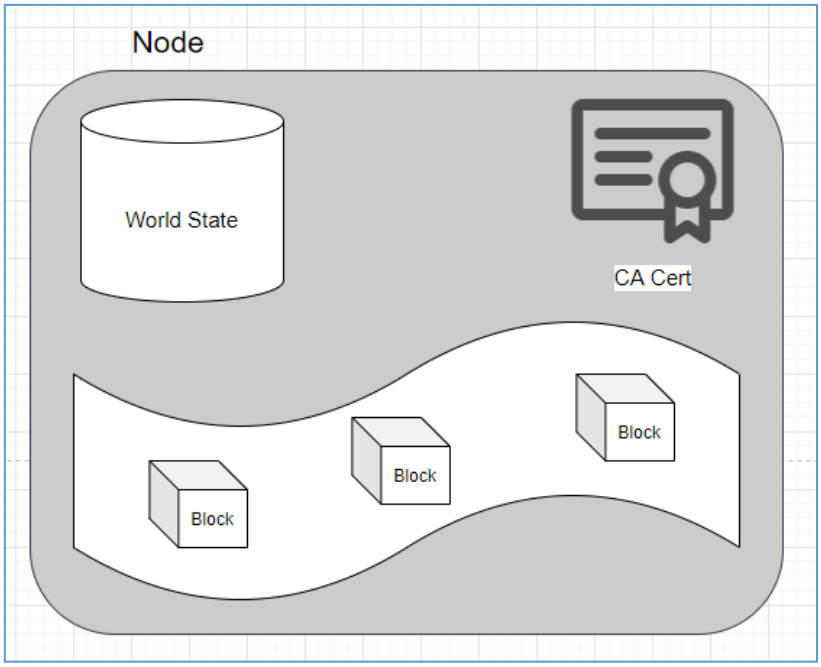



*Figure 22 - Hyperledger node components*

In Figure 22; a visual representation of a node in a Hyperledger network is shown. Each node contains the current values of all ledger states, or in other words all values are verified and have yet to be verified, CA cert which is security credentials for the nodes privileges to access the blockchain and finally the blockchain where all verified transactions are contained.

### 3.3.2. Flowchart

A flowchart represents the flow of operations from a high-level perspective. This flowchart represents a very basic user interaction with this application (cf. Figure 23) as it displays the flow of the program through the service user's perspective, showing the different interactions and options available for the service user. This flow chart does not show the complex interactions between the application layer and the data layer as this would be too complex for a high-level explanation of the systems operations.

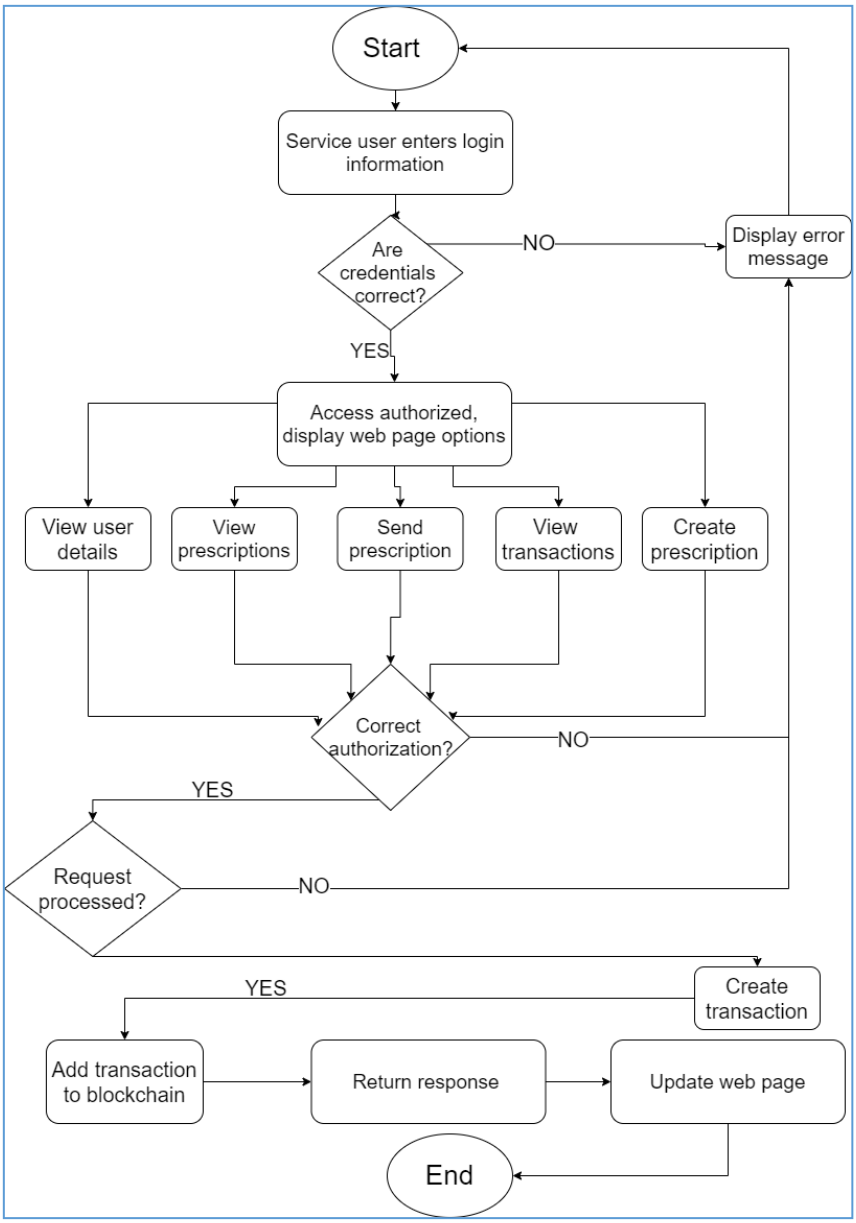



*Figure 23 - Flowchart of user's interactions*

### 3.3.3. Requirements table

The requirements table is broken down into phases, each phase is related to a sprint planned.

| ID | Name | Description | Phase |
|---|---|---|---|
| 1 | Blockchain environment | Setup test environment with blockchain. | 1 |
| 2 | Blockchain ledger details | Setup the details needed by each block of data in the blockchain ledger. | 1 |
| 3 | Blockchain connection | Setup API connection with Java application. | 1 |
| 4 | Java object classes | Setup Java classes for data to be on the blockchain ledger | 2 |
| 5 | Java blockchain functionality | Create methods for inserting things into the blockchain. | 2 |
| 6 | Java connection | Create a Java class to receive HTTP requests and to send HTTP responses. | 2 |
| 7 | Website setup | Create a functional JavaScript based website with aspects of HTML and CSS which connects to the Java application using API calls | 3 |
| 8 | Website view user details | Create webpage that allows users to view their details using API calls | 3 |
| 9 | Website view prescription history | Create a webpage that allows users to view the history of their prescriptions. | 3 |
| 10 | Website send prescription | Create a webpage that allows users to send a prescription to another user | 3 |
| 11 | Website view prescriptions | Create a webpage that allows users to see the prescriptions they currently own | 3 |
| 12 | Website create prescription | Create a webpage that allows the creation of a prescription | 3 |

*Table 4 - Requirements table*

## 3.4. Presentation layer

This section describes the design and planning of the presentation layer. Wireframes were used to show prototypes of the webpage and examples of what the finished product could look like. Use case diagrams were used to show clear interactions between each actor, in this case service users, GPs and pharmacists.

### 3.4.1. Key Screens

This section displays and explains prototypes that were designed for the presentation layer of this application. These screen prototypes were designed in Proto.io which is a prototyping software for websites. Five basic screens were designed which are displayed and described in this section.

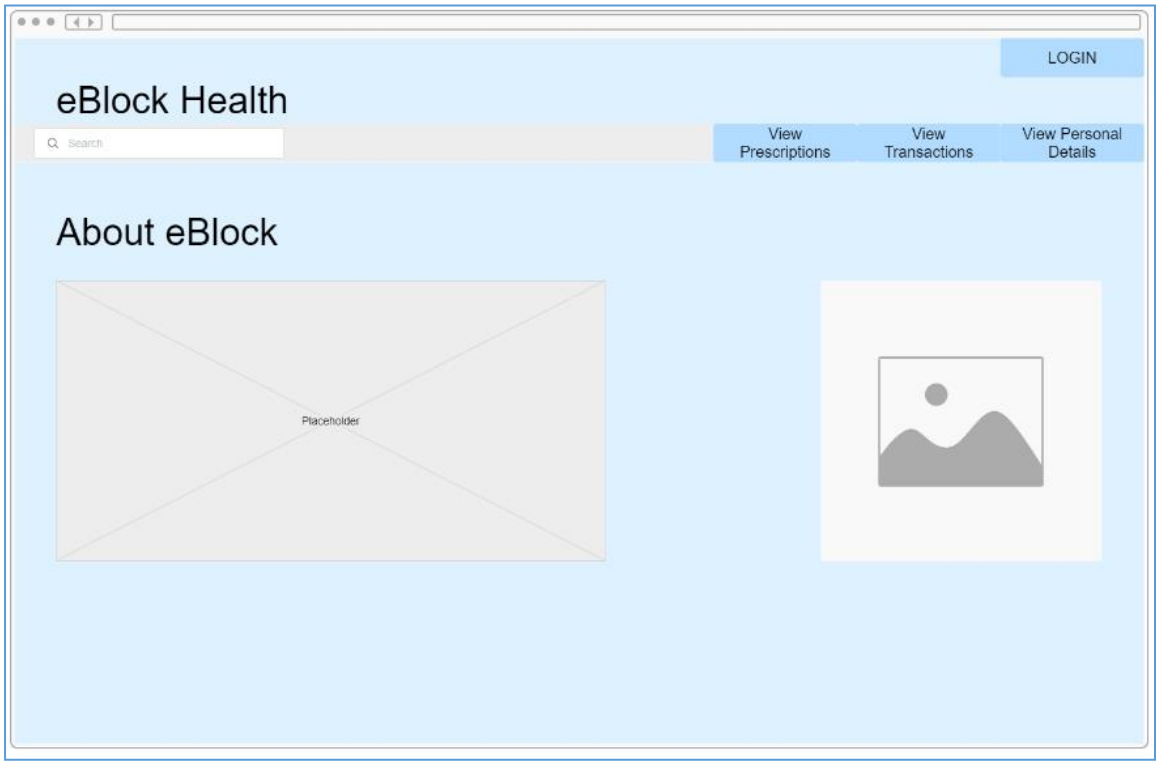



*Figure 24 - Landing page/ Home Page*

The home page or landing page is the first page that users will land on when opening this website. This page shows information (cf. Figure 24) about the Prescribing application and gives access to login, view prescriptions, view prescription history and view personal details. Although these functionalities cannot be accessed until a user has logged in. This will be prompted when a user tries to select any of these options.

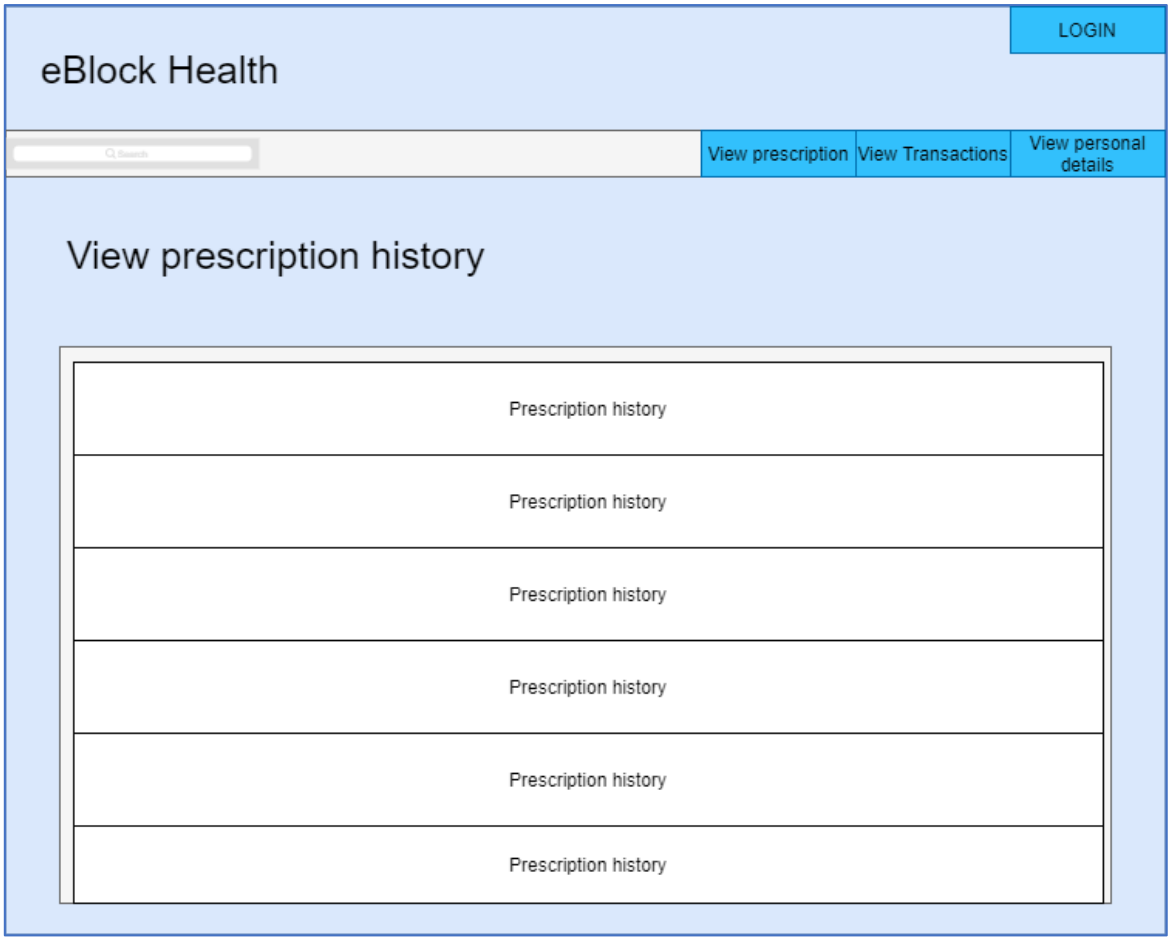



*Figure 25 - View prescription history page*

The prescription history page allows users to view the history of each prescription. This prescription history will include all details about the prescriptions and the wallet addresses of the parties involved. This page will contain a simple table of history as seen in Figure 25. Users can easily navigate through the website using the navigation bar which gives links for the main features of the application, which can be seen in Figure in 26

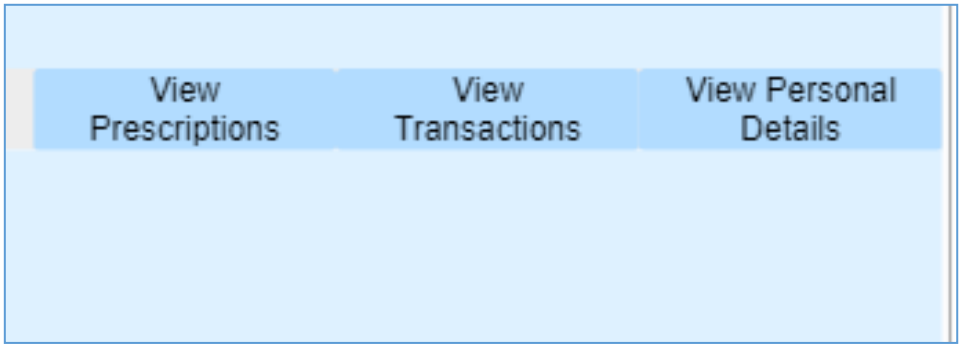



*Figure 26 - Navigation bar*

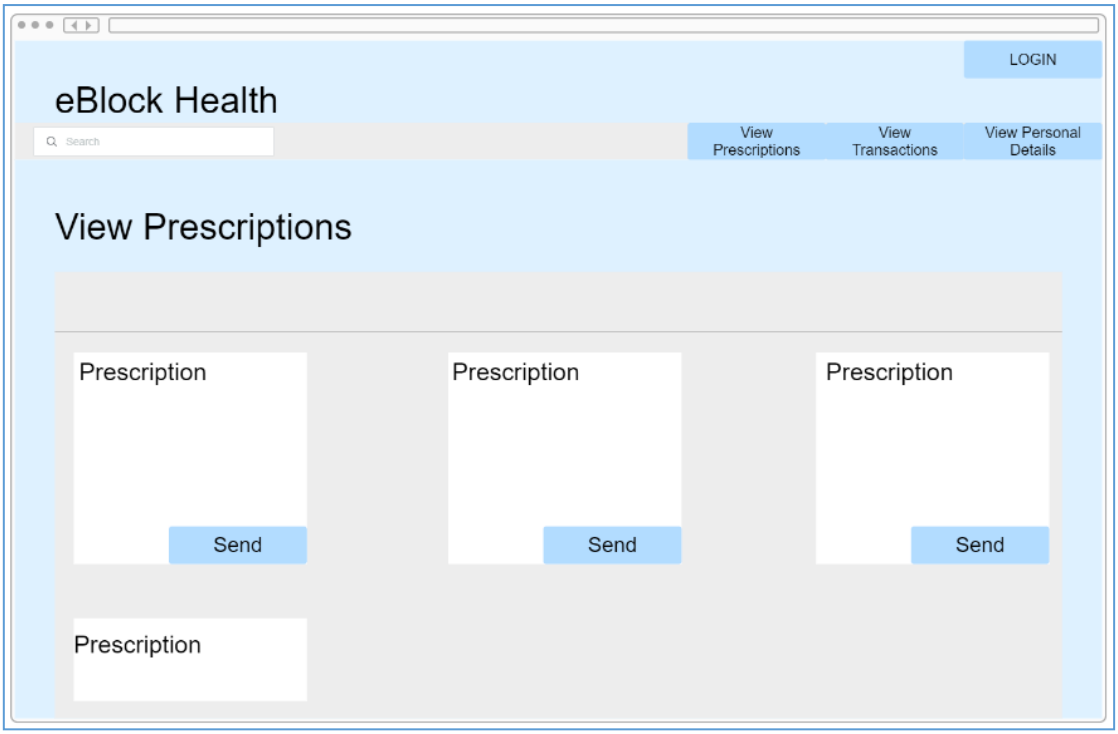



*Figure 27 - View prescriptions page*

The view prescriptions page allows users to view the prescriptions that they are currently in possession of. Each prescription is presented in a clear concise table with details of the prescription revealed when they select the prescription, which can be seen in Figure 27. Each prescription has the option to send the prescription to another address this is seen in Figure 28.

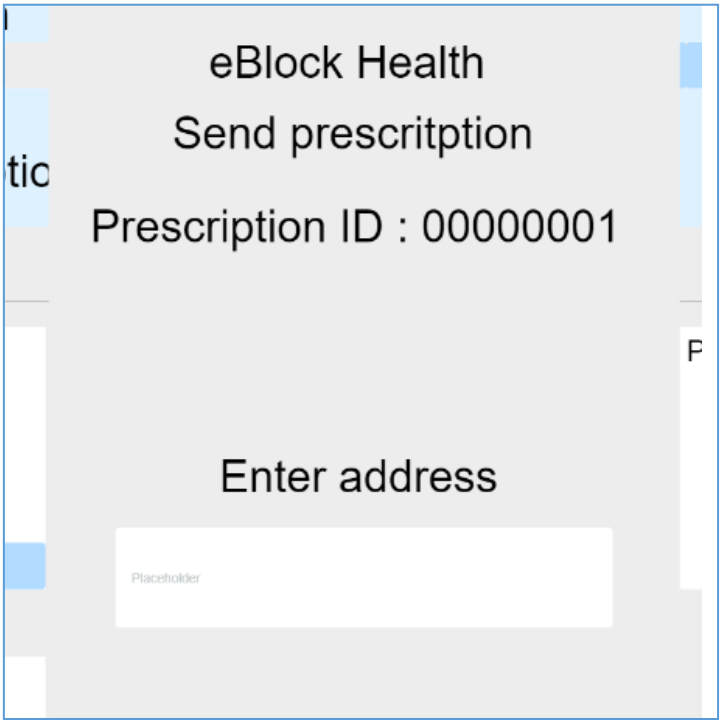



*Figure 28- Sending a prescription*

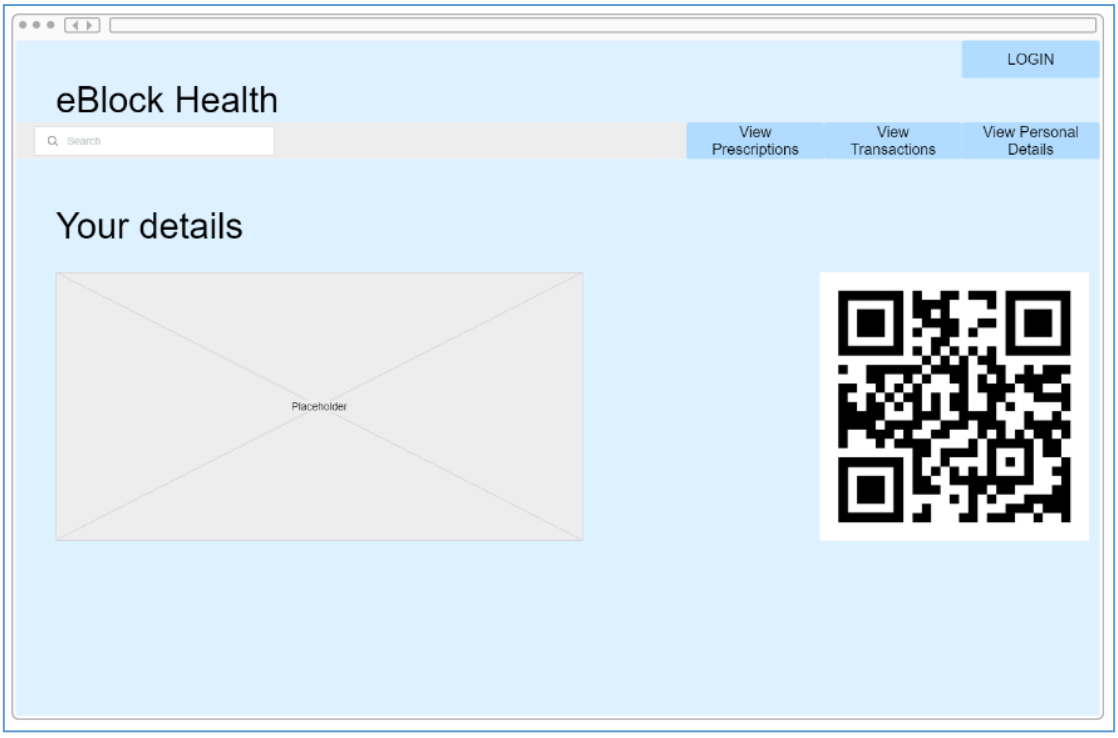



*Figure 29 - Personal details page*

A user can access their personal details within the website, from here they can also find their wallet address or the address that GPs can send their prescriptions to. This address can be displayed as both a QR code for the easy scanning of their wallet address and regular text to be copied (cf. Figure 29).

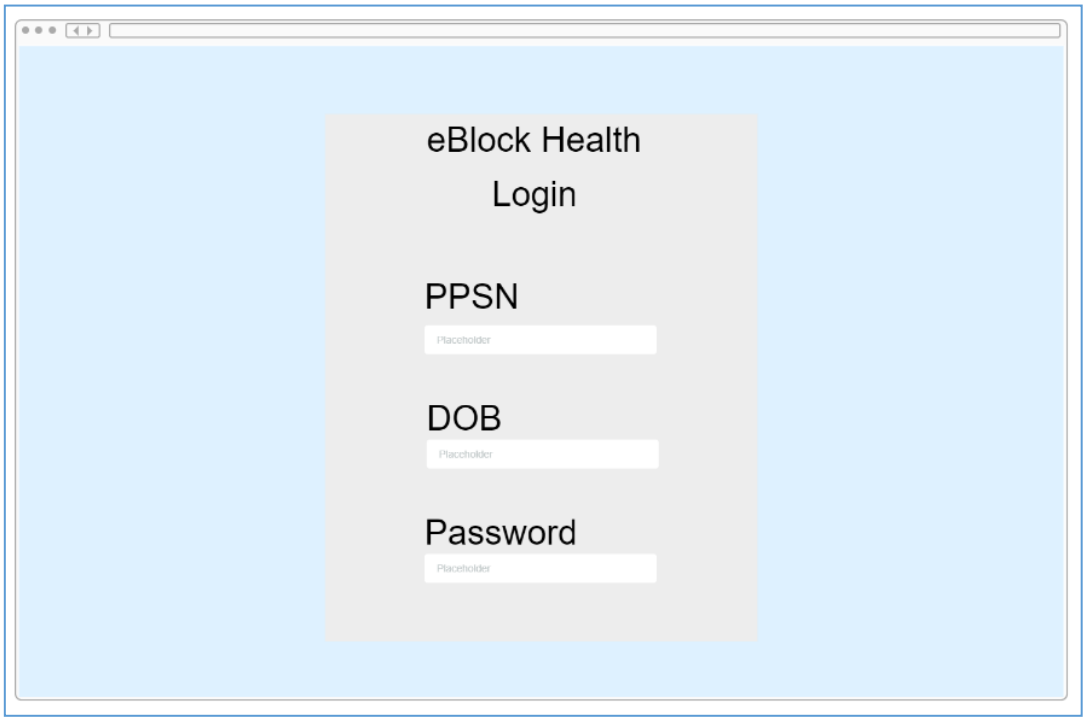



*Figure 30 - Login page*

The login page allows the user to access their personal information and their personal prescriptions. It takes the users PPSN, Date of Birth (DOB) and their unique password (cf. Figure 30). These accounts would be set up by a third party and their identity verified. The login page is accessed through the login button located on the top right hand of the page as seen in Figure 31. The login screen will also be prompted when the user tries to access pages that need authentication.

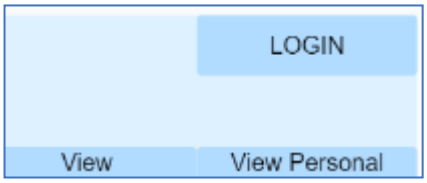



*Figure 31- Login button*

### 3.4.2. Use case diagrams

Use case diagrams are an effective way of describing the interactions between different users and applications, use cases are a powerful way of design and describing the requirements needed. The use case diagram identifies the actor, or the types of users involved with the application and what permissions each actor has.

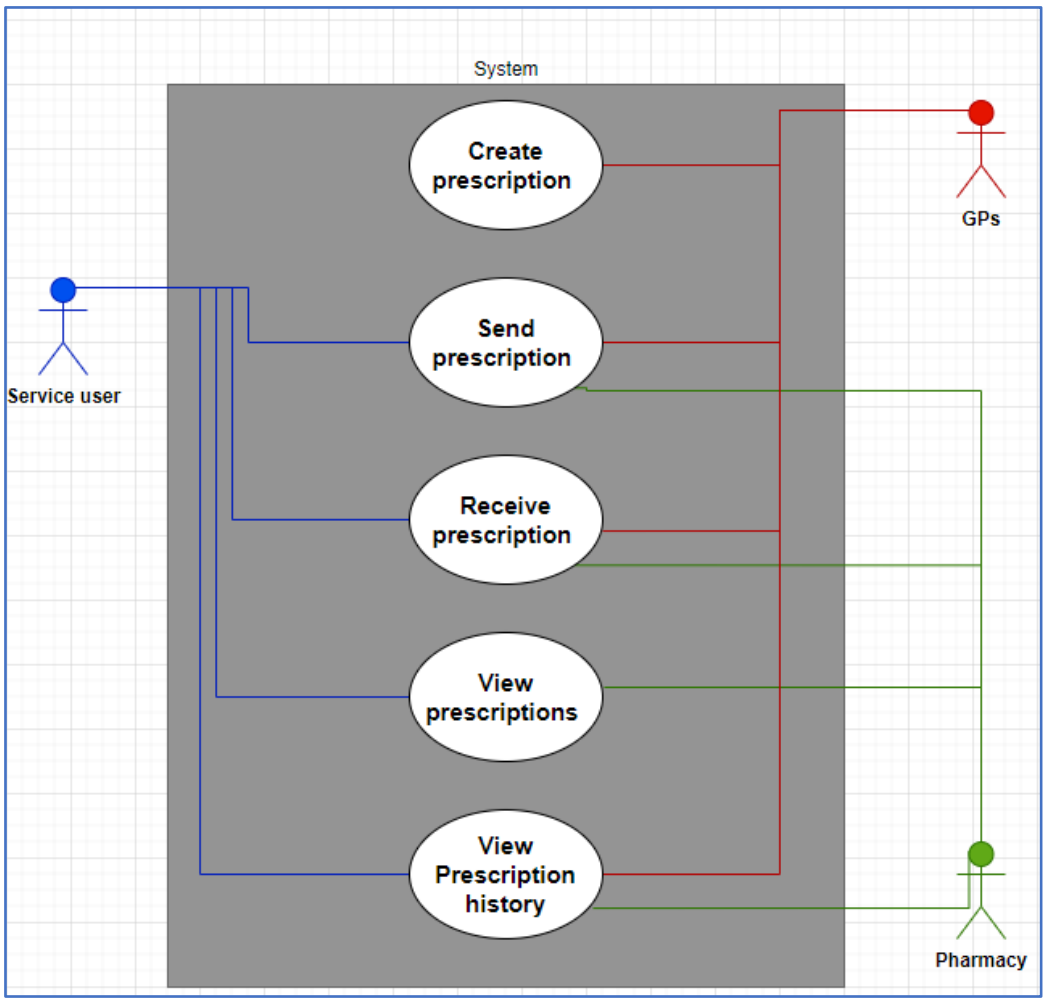



*Figure 32 - Use case diagram 1st iteration*

In the first iteration of the design process, 3 actors were identified: the service user which is the client or patient receiving the prescription, the GP or other medical practitioner and then finally the pharmacy or pharmacist as seen in Figure 32. Each of these actors have slightly different permissions and abilities within this system. Both the service user and the pharmacy have the same abilities to send, receive and view prescriptions whilst also having the ability to view transactions that have occurred. GP's have two differences to the previous actors; GP's can create a prescription but are unable to view prescriptions.

This diagram provided a general overview of the interaction between each actor and the system and the main functions of the application.

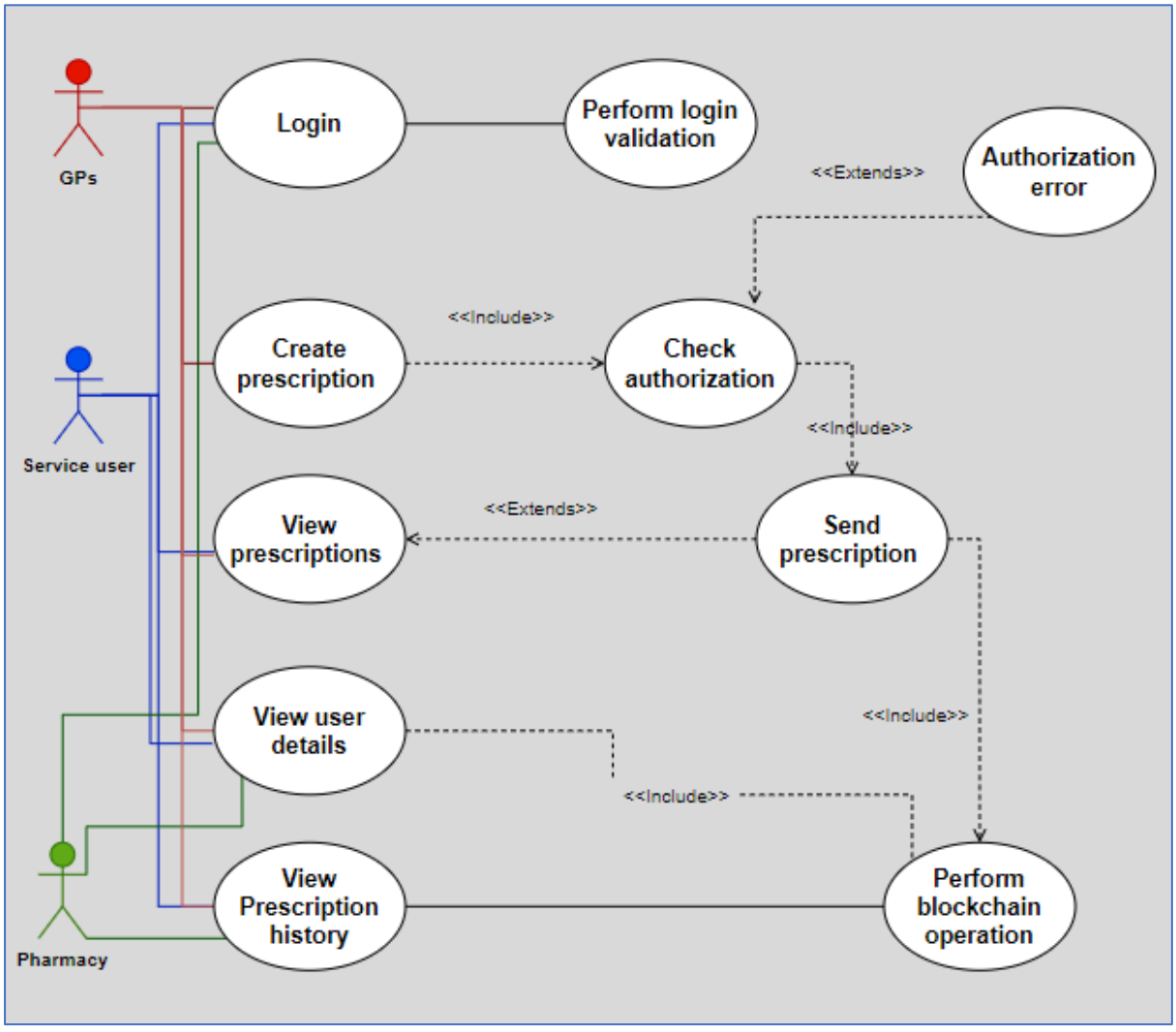



*Figure 33 - Use case diagram 2nd iteration*

The second iteration of the use case diagram provided a more comprehensive and detailed analysis of the system. While 3 actors remain, technically only 2 are needed. These are the service user and the GP, as both the service user and pharmacy have the same access rights. The 3 actors remain for clarification and what permissions each actor has. The addition of the login use case has been added, along with the login validation use case. While the 5 key features from the first diagram remain, they have been reshuffled to meet the flow of the

application more as seen in Figure 33. Send and receive prescriptions are now extensions of other use cases. The use cases; Check authorization, Authorization error and perform blockchain operation are all new use cases added to provide more detail on the processes of the system.

### 3.4.3. Conclusion

This section provided the detailed analysis of the presentation layer of this system using use case diagrams and a collection of wireframes of key screens that are displayed on the webpage. Clear program flow was established with use case diagrams which also identified the actors present in the application and the number of important use cases analysed.

## 3.5. Application layer

The application layer or the logic layer controls the rationale of a system; in this system, a Java application is used to provide functionality for the rest of the system. This layer provides an intermediary between the presentation layer and data layer; this approach allows for separation of concerns as the user does not directly interact with the data layer (80).

### 3.5.1. Model view controller

Model view controller (MVC) is an architectural pattern which has three separate components which each handle specific aspects of the application (81).

- The model handles the data transferred between the view and controller, i.e. a user class will have attributes that can be matched with each attribute found in the user table in the database
- The view component is what is used to control what the user wants from the application. In this project API's are used to recognise what the user is requesting from the webpage.
- The controller is the interface which handles the logic needed to fulfil the requests made by the user, it is an interface made between the view and the models. It usually involves tasks such as update, insert and delete.

A representation of the relationships between each component can be seen in Figure 34. The MVC architecture style is very popular among web-based development and provides a simplistic approach to developing the application layer.

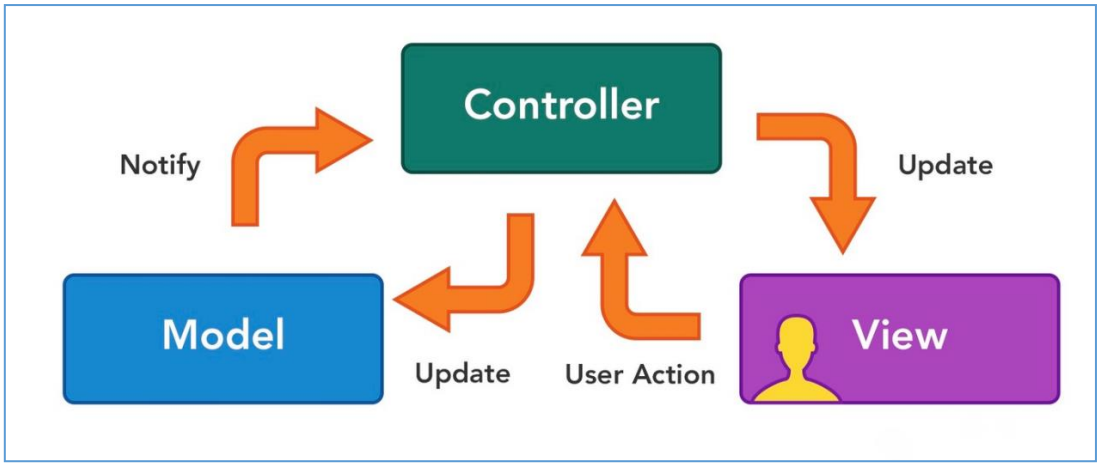



*Figure 34 - Example of MVC component relationships*

### 3.5.2. RESTful architecture

REST stands for Representational State Transfer and allows resources to be shared using URIs. Resources are shared using a stateless communication protocol such as HTTP (82). The key principles of REST architecture are:

1. "Application state and functionality are abstracted into resources. Any information, which is offered by the system and can be named, is possible to be represented by a 'resource'. Any concept that needs to be addressed, referenced and accessed must fit within the definition of a resource (83)."
2. "Resources must be uniquely identified and addressable using a universal syntax, such as a Universal Resource Identifier (URI) used in HTTP (83)."
3. "A uniform interface is shared by all resources for the transfer of state between client and the server. The set of operations, as well as supported content types, need to be well defined. At the same time, code on demand (such as JavaScript) could be optionally supported (83)."
4. "The communication protocol between the resource data provider and consumer has to be client-server, stateless, layered and cache enable. When the REST architectural principles are applied, as a whole, they provide enhanced scalability, generality of interfaces, independent deployment, reduced interaction latency and they can encapsulate legacy systems. With the advantages and characters of the REST, REST WS are broadly applied in system integration in most of the research fields (83)."

Rest was a key architectural component of the application layer as it is connecting the presentation layer to the functionality of the application layer to display the data and resources sourced from the ledger. An example of the REST process is seen in Figure 35. The REST architecture was implemented using JAX-RS which is discussed in Section 3.5.3.

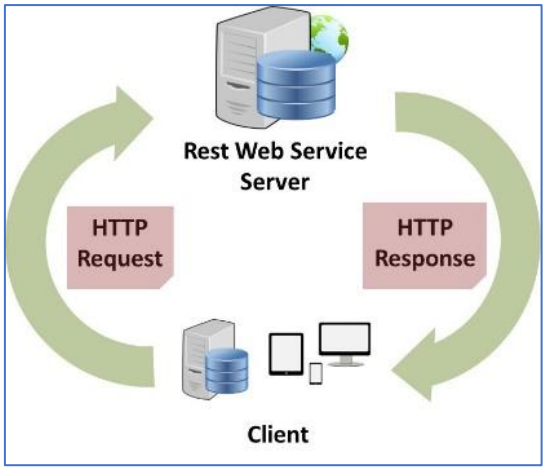



*Figure 35 - RESTful architecture*

### 3.5.3. JAX-RS

As mentioned in Section 3.3.1., the Java application will contain API endpoints to allow communication between the presentation layer and the Java application. These endpoints are set up using JAX-RS which is a built in Java library which allows Java methods to be called from the presentation layer using URI's an example is shown in Figure 35.

```
@Path("/users/{username}")
class Sample {

    public String getUser(@PathParam("username") String userName){
        //Process request
    }
```

*Figure 36 - Example of Java class using @Path notation*

### 3.5.4. Hyperledger API

Hyperledger fabric and Indy both provide SDK for Java applications which include API for accessing the Hyperledger blockchain. This API operates similarly to how the previous API operates (cf. 3.5.3). It establishes a connection between the Java application and the Hyperledger network and allows the Java application to commit transactions to the network. An example can be seen in Figure 37.

```
public void hyperledger(){
    try(Gateway gateway = builder.connect()){
        //Obtain a smart contract deployed on the network
        Network network = gateway.getNetwork("mychannel");
        Contract contract = network.getContract("fabcar");

        //Submit transactions that store state to the ledger
        byte[] createCarResult = contract.createTransaction("createCar")
        .submit("CAR10", "VM", "Polo", "Grey", "Mary");
        System.out.println(new String(createCarResult, StandardCharsets.UTF_8));
    }
```

*Figure 37 - Example of Java class executing Hyperledger transaction*

### 3.5.5. Conclusion

In this section the architecture of the application layer and the libraries needed were explained. The application layer consists of an MVC styled architecture that allows a simplistic approach to the application layer. The use of different API's allows the integrate easily with the different external applications needed for the whole project.

## 3.6. Data layer

This section describes the data layer architecture and design chosen for this ePrescribing application and an overview of the blockchain data structure. This analysis is shown using Entity Relationship Diagrams (ERD) and Interaction Sequence Diagrams (ISD).

### 3.6.1. Entity Relationship Diagrams

ERD's are a design diagram used to illustrate the relationships between identified classes or 'Entities' found within data storage applications such as Databases (84). In this application the blockchain framework Hyperledger is used. ERD's are composed of different entities with each entity containing attributes that describe the entity and then each entity in the diagram displays a relationship between other entities.

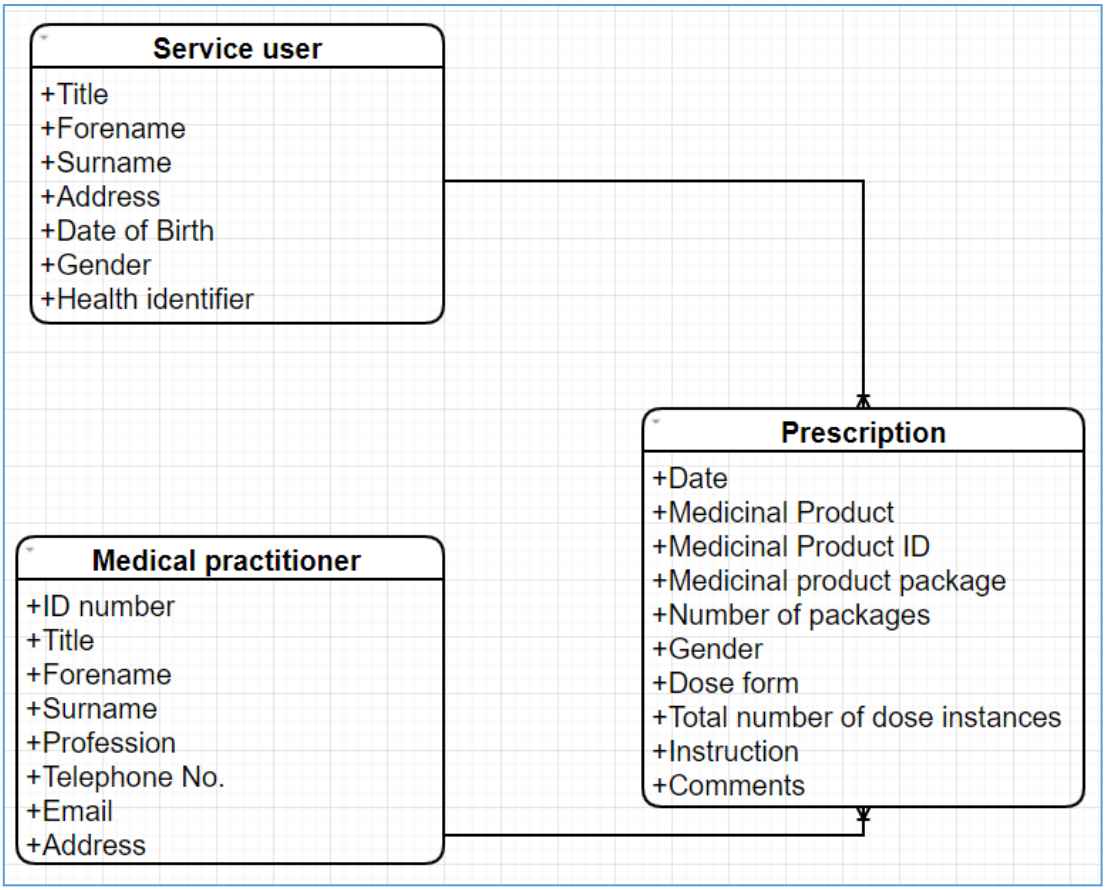



*Figure 38 - Initial design of ERD*

For the first design of the ERD components from Table 1, 2, 3 found in Section 2.5.4 were used to identify the information and entities needed for an ePrescribing application. The relationships between each entity are shown in Figure 38. This design provided a glimpse of how data would be stored in the blockchain but however there were parts missing such as transactions between entities and pharmacies.

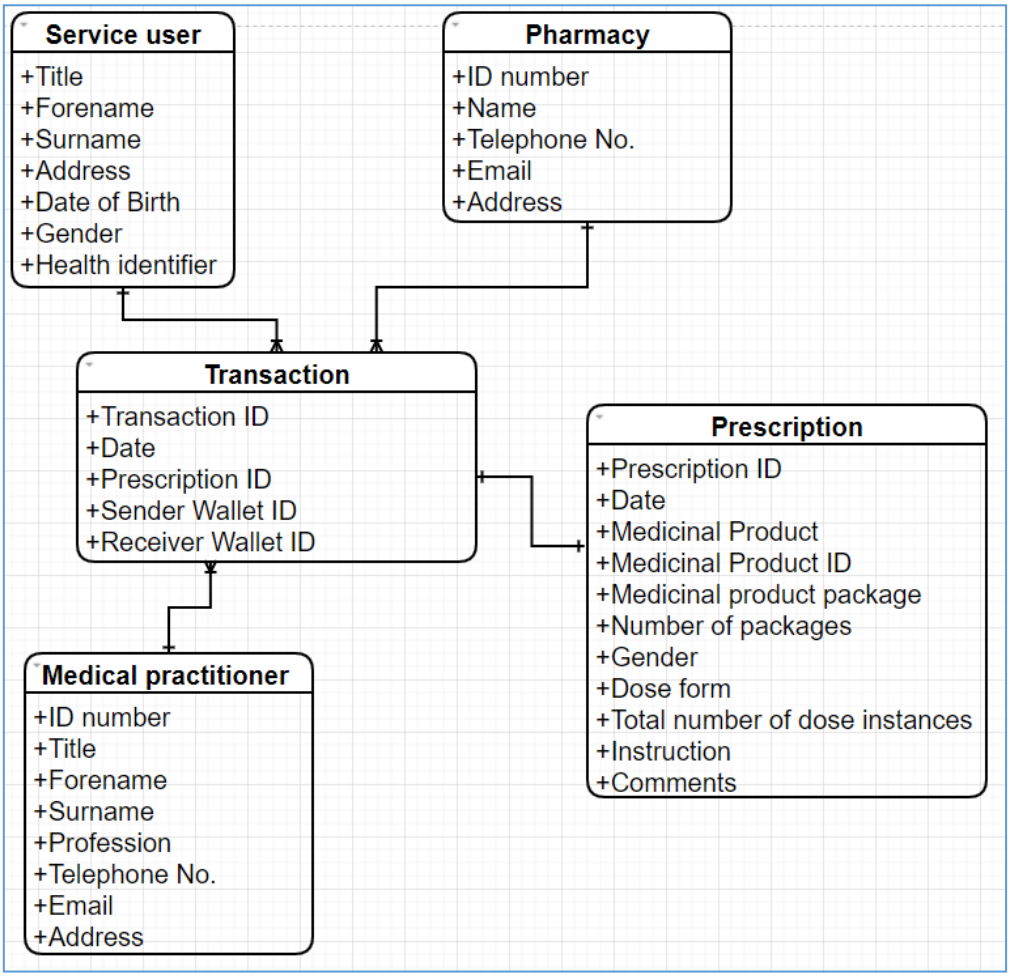



*Figure 39 - Second iteration of ERD*

The second iteration of ERD provides a more comprehensive description of the entities and their relationships as seen in Figure 39. The initial 3 entities, Medical practitioner, Prescription and Service user remain with the edition of the Transaction and Pharmacy entities. Each prescription is tied to a transaction as each prescription created must be linked with a transaction that occurs on the blockchain. Pharmacy is a key actor identified in chapter 3.4.2 and is included in this design.

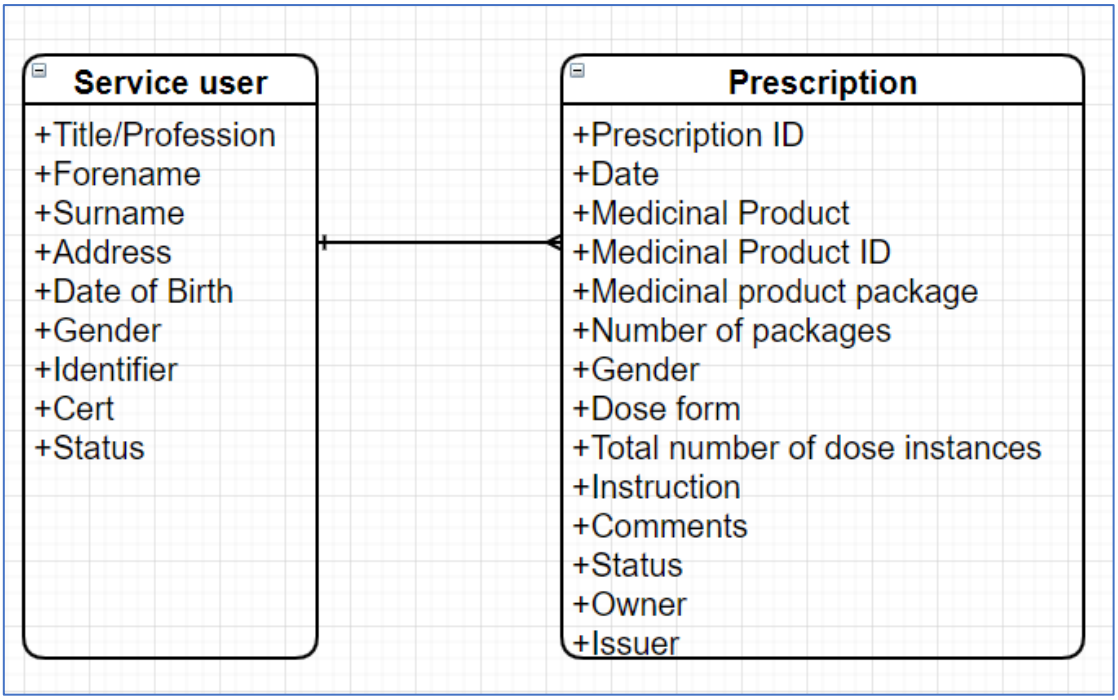



*Figure 40 - Third iteration ERD*

The third iteration of the ERD minimised the entities down into two entities, the ServiceUser and Prescription. As many of the details between each of the actors, Pharmacy, Service User and Medical Practitioner were similar it was decided that one entity would be made that share attributes of all of them. This simplified the relationships within the network and makes it easier to understand. The new ServiceUser entity also contains two extra fields, Cert and Status. The cert field specifies the public key associated with the user while the status specifies the role of the user from the three actors. The prescription entity also was modified to include three more fields: status the current availability of the prescription, owner the certificate of the user who currently owns the prescription and the Issuer the service user who created the prescription. This ERD is seen in Figure 40.

### 3.6.2. Interaction Sequence Diagrams

ISD is another useful way of describing the interactions between entities with regards to the data layers. ISD consists of objects and the messages sent to each object during a specified use case.

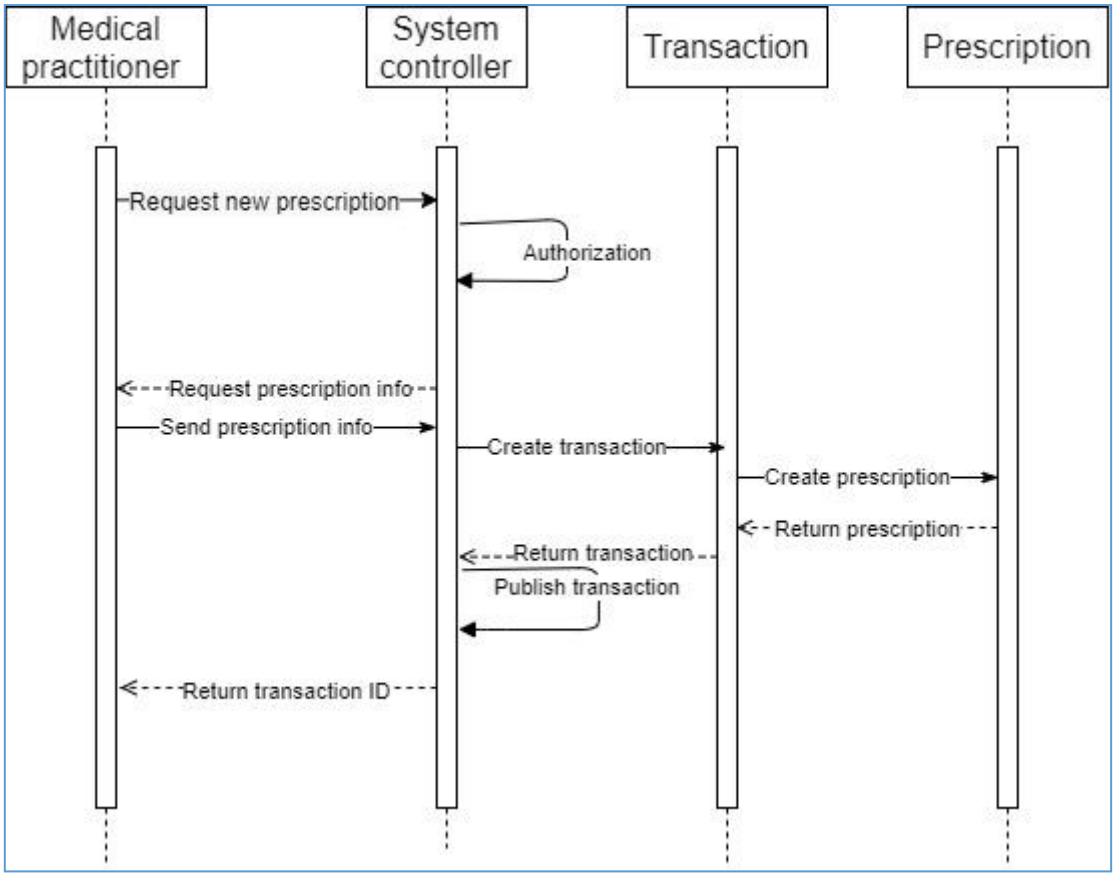



*Figure 41 - ISD for creating prescription*

The ISD pictured in Figure 41 represents the sequence of actions that occur when a GP or medical practitioner creates a new prescription for a service user. This ISD displays 4 main objects present in this exchange which are the GP themselves, the system controller which handles the logic of the request, the transaction class which handles the data and the prescription which handles the data of the prescription described by the medical practitioner. Only medical practitioners can create prescriptions and authorization is needed as described in the use case diagram in chapter 3.4.2.

### 3.6.3. Conclusion

This section describes the architecture of the data layer present in this system, using different architectural diagrams such as ERD to show the relationship between different entities and ISD to show how different objects work together in a specific use case.

## 3.7. Conclusions

In this chapter, the design of this ePrescribing system was presented. Different methodologies were discussed to decide which would be appropriate to the scope of the project, Scrum; an Agile based methodology was chosen as it is dynamic and is flexible to projects with ever changing requirements. Test-Driven development characteristics are also used to improve the quality of code.

Following this, the technical architecture of the system was presented, and in this case, the 3-tiered architecture was discussed and the details of each of the tiers was briefly summarised. The requirements identified for this project were also identified and the sprint for each requirement was selected. The tiers identified were also discussed in detail.

The presentation layer was presented which displayed different wireframes and visual prototypes, were to provide a representation of what the finished product might look like. Use case diagrams were also used to show interactions between the user and the application.

The application layer was presented and showed the architectural structure used and how API's are used to communicate with each of the different technologies linked with the application layer.

Finally, the data layer was discussed with Entity relationship diagrams and interaction sequence diagrams to show relationships between each entity and how they interact with each other to complete a use case.

# 4. System Development

## 4.1. Introduction

Chapter four presents the designs planned, the implementation choices taken through the development process and the development process previously discussed in the chapter 3. This chapter is presented in nine sections.

Section 4.2 – 4.3 will cover a detailed analysis of the environment used for this project and the software methodology used. Section 4.4 – 4.7 will discuss the development stages; the system architecture which includes the data layers, application structure and presentation. Section 4.8 – 4.9 discusses the challenges encountered when developing this project and the conclusion.

## 4.2. Software Methodology

As discussed in section 3.3.2, the methodology chosen for this project is Scrum which is based on the Agile methodology. Scrum consists of a backlog of requirements that need to be completed. These requirements are highlighted in section 3.3.3. Each sprint for this project lasted 2 weeks, with a total of 8 sprints needed to complete the development of the project. A scrum project management board on GitHub was used to organize these requirements. These sprints are highlighted in the tables 5 – 12.

| Sprint 1 | 23/11/2020 - 06/12/2020 |
|---|---|
| Task | Details |
| Create GitHub repository | Created a GitHub repository to track changes made to code. GitHub was also used to track issues and developments using the GitHub project management tool. |
| Investigate & Research into Hyperledger Fabric | Research into the structure and how Hyperledger fabric operates, the software and libraries needed to run the network. |
| Investigate & Research into Hyperledger Indy | Research into the structure and how Hyperledger Indy integrates with Fabric. Issues later occurred that halted Indy's use in this development. |
| Install prerequisites for Hyperledger Fabric | Docker, Hyperledger Fabric binary files, Linux subsystem. A Linux subsystem for windows was necessary to run the network. |

*Table 5 - Sprint 1*

| Sprint 2 | 07/12/2020 - 20/12/2020 |
|---|---|
| Task | Details |
| Created a test network for Hyperledger Fabric. | Created a sample blockchain network using Hyperledger Fabric documentation. |
| Modified test network to fit requirements. | Modified the test network to suit the requirements of the implementation. The network contains 2 certificate authorities and a database to store the world state and the data collected. |
| Created shell scripts to start, stop and assign variables to the network. | Created shell scripts to start-stop network and add environment variables easily. Unlike Windows, Linux sets temporary environmental variables which caused issues with the execution of the network. |

*Table 6 - Sprint 2*

| Sprint 3 | 28/11/2020 -10/01/2021 |
|---|---|
| Task | Details |
| Research into developing chaincode for the network. | Researched using the Hyperledger fabric documentation and example chaincode. Research proved difficult as many examples of the chaincode were written in the language JavaScript but chaincode in Java was necessary. |
| Created prescription chaincode and prescription model. | Created a Gradle java app that can add, update and view prescriptions. The prescription model was also created which contains the attributes of the prescription. Gradle java application was necessary as the project structure of a maven project cannot integrate with the structure necessary for the blockchain network. |
| Created script for deploying chaincode to the network | Created a shell script for deploying chaincode to the network automatically when the network is started. |

*Table 7 - Sprint 3*

| Sprint 4 | 11/01/2021 - 24/01/2021 |
|---|---|
| Task | Details |
| Implemented monitor script to track changes and outputs made by the network. | Altered and implemented the Fabric script to monitor traffic from the network, including inserted data onto the blockchain. |

| | |
|---|---|
| Researched into development of applications that could utilize chaincode. | Research into Hyperledger's documentation regarding development of applications to use chaincode; again much of the documentation available contained information regarding JavaScript. |
| Created Gradle java web app | Created a Gradle java web app, closely followed the sample java app provided by Hyperledger with regards to imports and structure. |

*Table 8 - Sprint 4*

| Sprint 5 | 25/01/2021 - 07/02/2021 |
|---|---|
| Task | Details |
| Created methods to connect to the blockchain network. | Created a method for connecting to the blockchain network and accessing the chaincode. Many issues occurred when connecting to the network due to location, file names of the chaincode and the authorization of the chaincode on the network. |
| Created methods for creating a new entity or user access to the network. | Created methods which create a wallet which stores a user's certificates for accessing the network. |
| Created methods for creating a new prescription, editing prescription, and viewing prescriptions. | Created the main methods for access prescription data and manipulating data on the blockchain. |
| Created methods to transfer a prescription and update prescription status. | Created additional methods for altering specific data only in a prescription. |

*Table 9 - Sprint 5*

| Sprint 6 | 08/02/2021 - 21/02/2021 |
|---|---|
| Task | Details |
| Created template website with the web pages: Login, user details, created prescription, view prescriptions and view prescription history. | Created the foundation website where each functionality would be expanded on. Webpages were made using bootstrap, html, and CSS. |
| Identified methods in the java web app needed by the website and created API paths for each of them. | Created URL paths for each method in the java web app needed by the website to provide resources for the web page. Issues with post requests as they were restricted by tomcat and had to be enabled. |
| Deployed java web to tomcat server. | Deployed the java web app to tomcat server, this involved changing directories and locations associated with the network to reflect the location on the server. |
| Created server utility to send requests to the server. | Created a JavaScript/Jquery server file that allows http |

| | |
|---|---|
| | requests to be sent to the java web app deployed on the tomcat server. |
| Created script to compile and deploy application to tomcat server. | Created a shell script which copies the html/js files needed for the webserver, compiles the java web app into a war, copies docker files necessary to recognise the network on the server and then copies the war file onto the server. |

*Table 10 - Sprint 6*

| Sprint 7 | 22/02/2021 - 07/03/2021 |
|---|---|
| Task | Details |
| Created user details chaincode. | Created another Gradle java application to store user details which access the network, this application contained methods to create a user, get details of the user and update a user's details onto the blockchain. |
| Created session functionality and created login authentication with server. | Implemented a http session when a user logs in that stores a user's details and authenticates it with java web app and the network. |
| Created check session and invalidate session. | Created functionality to check if a user is logged in before accessing the webpage and if they have the correct permissions to view certain content. Added a logout button that removes their details and information from the server. |
| Created functionality to view the user's personal details and certificate associated with that account. | Created the functionality to view a user's own personal details and associated public key certificate with their account. |

*Table 11 - Sprint 7*

| Sprint 8 | 08/03/2021 - 21/03/2021 |
|---|---|
| Task | Details |
| Created functionality to create a prescription | Created functionality to create a prescription from a set of predefined prescriptions. |
| Created functionality to view prescriptions and transfer prescriptions. | Created the functionality to view prescriptions associated with the public key certificate associated with the user, created the ability to send prescriptions to another user. |
| Enabled SSL | Enabled SSL security with Tomcat so requests were encrypted. |

| | |
|---|---|
| Created functionality to view prescription history. | Created the functionality to view the history of prescriptions. This includes previous owners of the prescriptions and any details that may have been altered. |
| Added design elements and display features. | Created notifications to website and style elements to the website. Styled the website to have a simple and clear to understand design. |
| Refactored code. | Refactored code and verified all components were clear to understand. Commented code for difficult sections to understand. This process was completed on both the chaincode, shell scripts, JavaScript, HTML, CSS and Java web app code. |

*Table 12 - Sprint 8*

## 4.3. Environment

Prior to the development of this project, the environment for the project had to be built. This involved initialising a Linux subsystem, creating a GitHub repository and a local workspace for the project. Along with downloading and installing the correct libraries, binaries and software needed for this project.

The first stage of the environment initialization process was to install a Linux operating system that would be capable of running the application and project. This installation required Windows Subsystem for Linux (WSL). WSL allows for a Linux environment used on a windows machine without the need to initialise a virtual machine (85). WSL was first enabled on the host system and after this Ubuntu 20.04 was installed. An Ubuntu Linux system was necessary for this project as some tools could only operate under a Linux environment, this will be discussed in detail in section 4.5.

The next stage was to create a Git repository to store and track changes made to the project, this was important as it provided version control of the project. In case of errors in the development process, the previous version of the project could be restored. GitHub also provided a method of project management as stages of the development could be mapped and tracked with the project management tool. When the GitHub repository was made, issues and features were documented on the project management board. The GitHub for this project can be viewed here https://github.com/daandymaan/BlockchainPrescribing .

The next stage was to create a directory structure and project workspace on the Linux system. The first step was to pull the Git repository onto a folder on the Linux system and from there the folder structure was created. Each main component of the project was isolated into a separate directory as found in Figure 42.

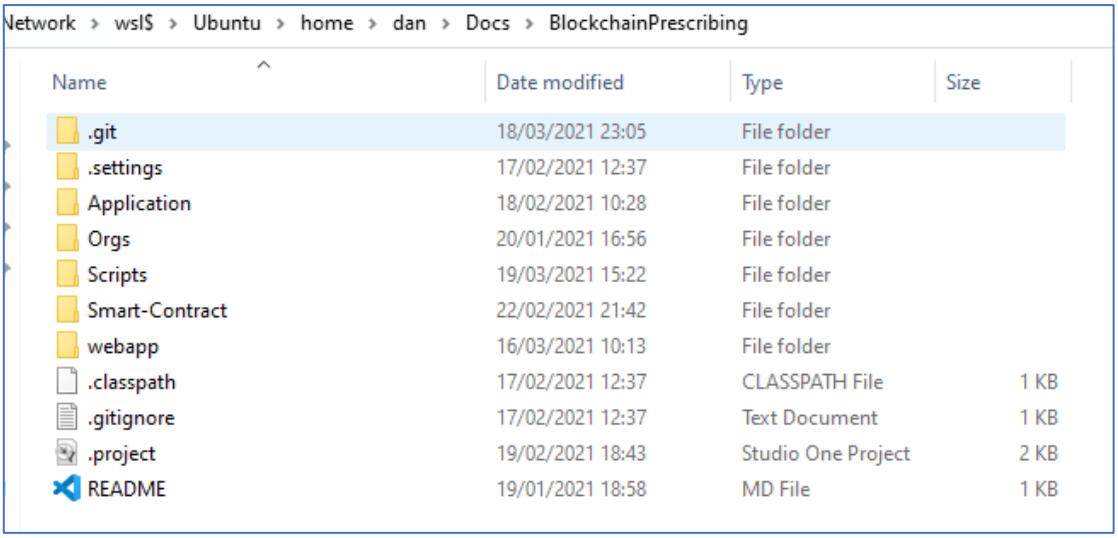



*Figure 42 - Workspace structure*

The five core folders of this project are the “Application”, “Orgs”, “Scripts”, “Smart-Contract” and “webapp”.

The Application folder is for files and resources relating to the Java web application or the application layer of the program.

The Orgs folder is for docker files relevant the connect to the network.

The Scripts folder is for shell scripts needed for the project, such as starting and stopping the blockchain network.

The Smart-Contract folder is for Java applications to be deployed as chaincode onto the blockchain which act as an interface between the web application and the blockchain. These Smart-Contracts make up a part of the data layer for this project.

The webapp folder is for HTML, CSS and JavaScript files relating to the website or the presentation layer of the application.

Once the project structure was set up, the appropriate tools and libraries that were needed for this project were downloaded and installed. This included: Java JDK version 1.8, Tomcat version 8.5, Visual studio code, Docker and Hyperledger fabric samples repository. The Hyperledger samples repository contained the core binaries needed to run a blockchain network whilst also contain samples of applications and code as well as documentation in how to operate with it. This documentation was researched and studied before development began on the project (86).

Tomcat also required further initialisation as by default the server application only authorized HTTP GET requests to and from the server.

After the initialisation of the environment and research into how the Hyperledger Fabric framework operated, development started on the data layer of the project.

## 4.4. Architecture Summary

As discussed in section 3.3, the system incorporated a three-tier architecture model. The blockchain system has a webpage on the presentation layer which communicates with Java web application hosted on tomcat server, sending requests to the Java web application for resources or to create resources. The Java web application then communicates with the Java chaincode deployed onto the blockchain network. The network then validates resources and returns the data or approves newly added data to the blockchain. The separation of concerns between each level of architecture makes identify issues within the project easier. A brief architectural summary of the complete project is seen in Figure 43.

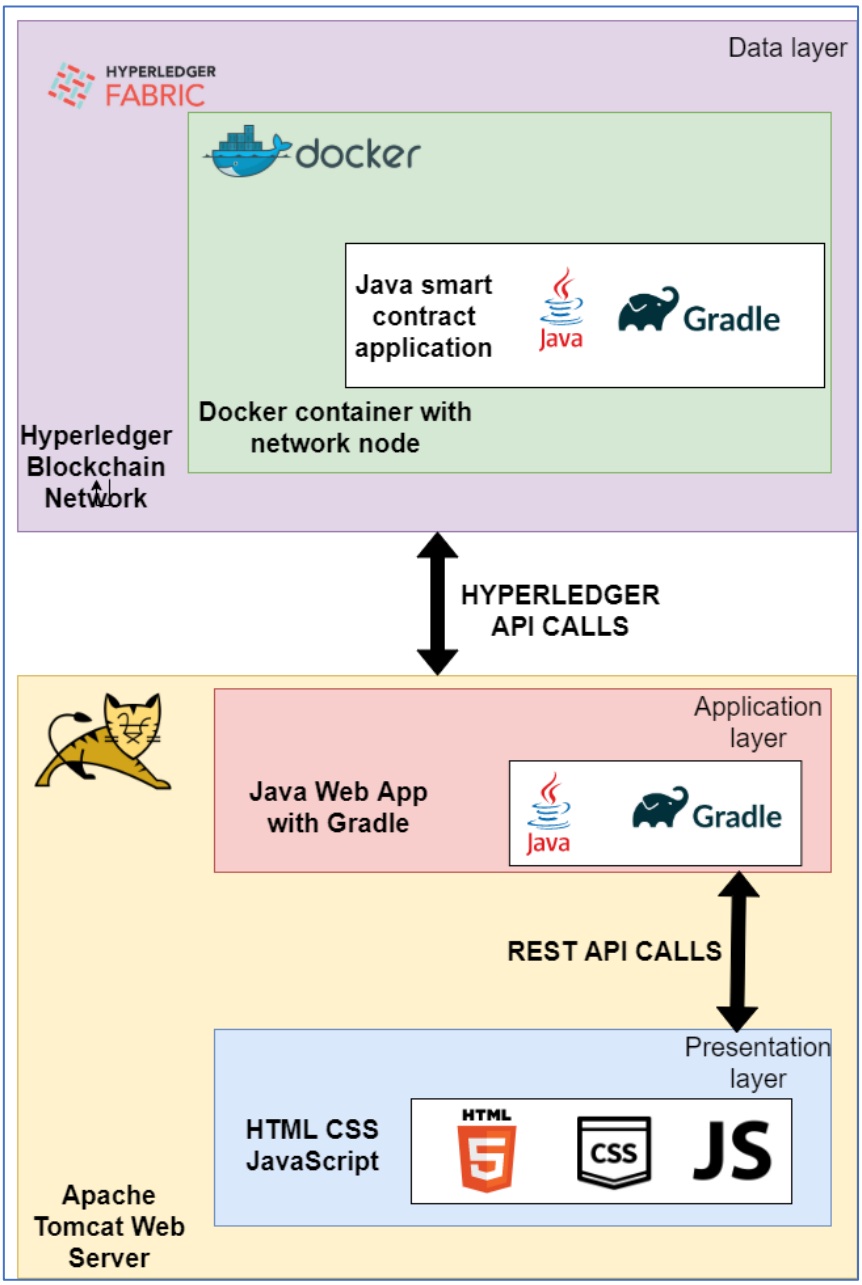



*Figure 43 - Architecture summary*

## 4.5. Data Layer Development

Section 4.5 discusses the first phase of the development process which included the development of the blockchain network itself and the smart contract code that was deployed to the network. It also discusses many of the shell scripts that were used to aid the development process and the logic and reasoning behind many of the choices. Finally the section briefly discusses the issues encountered in the development, which are discussed in more detail in section 4.8.

### 4.5.1 The Blockchain Network

The blockchain network in this project was created with a combination of docker containers along with the resources provided by the Hyperledger Fabric binary files and other resources. In this project, the sample script to generate the network was used and expanded on to fit the required needs of the system. The script allows for customization of the components of the network. In total the network is comprised of 8 docker containers as shown in Figure 44. The 8 containers are two network peers, one for each organisation, two CouchDB databases, one for each peer, a certificate authority (CA) for each organisation and finally an ordering node and an ordering CA.

- Peer: Peers are the fundamental component of the network; they store information from the blockchain and validate transactions before they are committed to the blockchain. They contain the relevant chaincode that is used to manage data on the ledger. In the project only 2 peers were used as to demonstrate the decentralisation and the approval functionality of the blockchain.
- CouchDB: CouchDB acts as a secondary database alongside the LevelDB database which is created by default when a peer is created. CouchDB gives query support for data values rather than keys of the asset. It can also support queries for large sets of data.
- Organisations: Organisations are key to the network's stability, and if an organisation exists in the network, the network will be operational. Three organisations were used, one for each peer and one for the ordering node, to demonstrate the decentralisation of the network and how different organisations react with each other as shown in Figure 42.
- Certificate Authority (CA): CA's are in each organisation and maintain and create identities that belong to each organisation. CA's are key to security as they limit access to only users who have valid certificate credentials to access the blockchain network.
- Ordering node: Ordering nodes control the order of the transactions on the ledger, they also have access controls from channels restricting who can read and write data and who can reconfigure them (86).

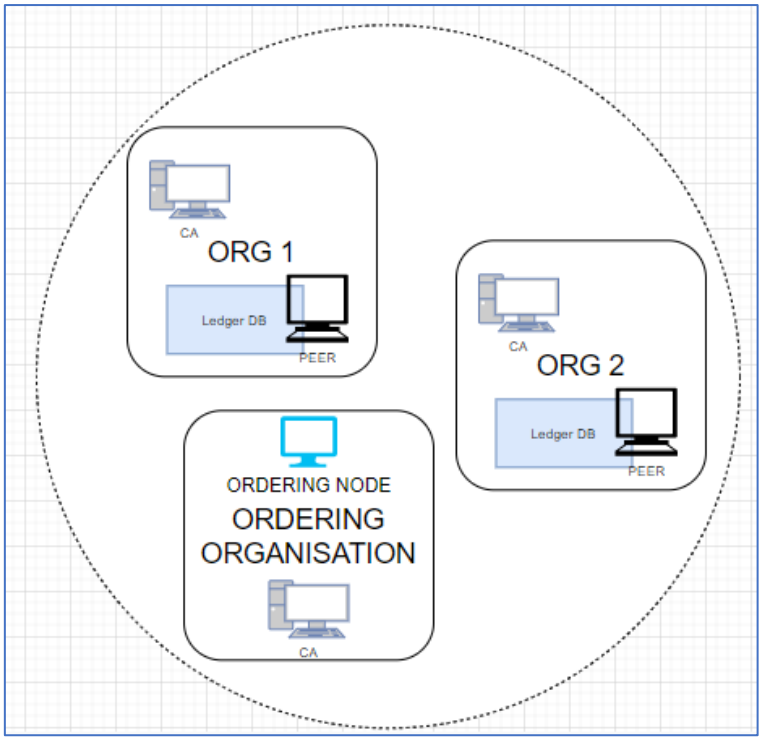



*Figure 44 - Network configuration*

Each of the components run in a docker container which are then connected in a docker network. The two peer organisations are used to connect to the network and add, update or view data on the ledger. These organisations details are stored in the "Orgs" folder pictured in section 4.3, Figure 42. The configurations allow applications to connect to the blockchain network. The docker files are created when the network is started with various shell scripts which is discussed in section 4.5.2.

### 4.5.2. Shell Scripting

There were a variety of different shell scripts used in the development of the blockchain application. The scripts helped manage the operation of many different components of the network and to help automate many tasks that were repetitive. All the shell scripts are in the "Scripts" folder on the workspace for this project. In this section the key scripts are discussed as there were a total of 6 scripts used during the development of this project.

#### 4.5.2.1. Start-network.sh

This script's functionality was to set the necessary environmental variables; start the network with the appropriate configurations, export the peer configurations and then finally deploy the chaincode. The network is started with the use of the inbuilt configuration script available with Hyperledger Fabric. The "stop-network.sh" script is also called in this script as to stop the network in case it is already operational. The chaincode is deployed using another script that was created for this project. This is seen in Figure 45.

start-network.sh

```
#!/bin/bash
#
function _exit(){
    printf "Exiting:%s\n" "$1"
    exit -1
}

#Returns the value of the last pipe which exited
set -o pipefail
#-e = Exits if command has non-zero status
#-v = Displays each command run by bash script
set -ev

#Current directory
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

#Export appropriate enviromental variables
export PATH=/home/dan/Docs/fabric-samples/bin:$PATH
export FABRIC_CFG_PATH=/home/dan/Docs/fabric-samples/config/
export PATH=$PATH:$FABRIC_CFG_PATH
export COMPOSE_PROJECT_NAME=net
export IMAGE_TAG=latest
export SYS_CHANNEL=system-channel

#Stop network
./stop-network.sh

#Go to network directory
cd /home/dan/Docs/fabric-samples/test-network

#Creates a network with ca and each have a couchdb
./network.sh up createChannel -ca -s couchdb

#Copy peer configurations to external folder
cp "/home/dan/Docs/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/connectio
cp "/home/dan/Docs/fabric-samples/test-network/organizations/peerOrganizations/org2.example.com/connectio

cd /home/dan/Docs/BlockchainPrescribing/Scripts

# Run this script to install chaincode
./deployCC.sh infocontract /home/dan/Docs/BlockchainPrescribing/Smart-Contract/info-contract
./deployCC.sh pc /home/dan/Docs/BlockchainPrescribing/Smart-Contract/java-contract
```

*Figure 45 - start-network.sh script (1)*

#### 4.5.2.2. Stop-network.sh

This script's functionality is to stop the network and remove any docker images and remnants of the old network. When each network is first created, a set of certificates are created for each peer to give them access to the network, these are unique and are created when the network is started. Any associated identities or certificates must be removed and regenerated. Each docker image must be stopped and removed when the network is restarted. This script also stops the tomcat server. This is seen in Figure 46.

stop-network.sh

```
function _exit(){
    printf "Exiting:%s\n" "$1"
    exit -1
}

#Returns the value of the last pipe which exited
set -o pipefail
#-e = Exits if command has non-zero status
#-v = Displays each command run by bash script
set -ev

#Current directory
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

#Export appropriate enviromental variables
export PATH=/home/dan/Docs/fabric-samples/bin:$PATH
export FABRIC_CFG_PATH=/home/dan/Docs/fabric-samples/config/
export PATH=$PATH:$FABRIC_CFG_PATH
export COMPOSE_PROJECT_NAME=net
export IMAGE_TAG=latest
export SYS_CHANNEL=system-channel

#Stops tomcat server
$CATALINA_HOME/bin/shutdown.sh

#Go to network directory
cd /home/dan/Docs/fabric-samples/test-network

#Remove docker
docker kill logspout || true

#Stop network if already running
./network.sh down

#Delete wallets/identities folder
if [ -d "/home/dan/Docs/BlockchainPrescribing/wallets" ]; then
    rm -r /home/dan/Docs/BlockchainPrescribing/wallets
    #docker rm $(docker ps -a -q)
fi

#Delete wallets/identities folder
if [ -d "/opt/tomcat/apache-tomcat-8.5.63/webapps/wallets" ]; then
    rm -r /opt/tomcat/apache-tomcat-8.5.63/webapps/wallets
    #docker rm $(docker ps -a -q)
fi
```

*Figure 46 - stop-network.sh script (1)*

#### 4.5.2.3. monitor.sh

This script monitors the activity from the network, it captures the logs created by the network which includes details about, starting up of the network, the transactions that occur on the network and the validations and approvals that are executed on the network. The script then displays these logs to the terminal window. This script was provided by Hyperledger Fabric.

#### 4.5.2.4. config-env.sh

This scripts functionality is to initialize the environmental variables associated with each of the peers. These environmental variables are used by many of the scripts in this project. This script was also created and provided by Hyperledger Fabric.

#### 4.5.2.5. warDeploy.sh

The warDeploy scripts functionality is not related to the network or the blockchain, this script prepares and compiles the resources for the Tomcat server. It first stops the Tomcat server and copies the peer configurations into the resources folder in the Java web application. It then copies the contents of the webapp folder to the webapp folder in the Java web application. It removes the previous version of the war file deployed on the Tomcat server. Hereafter it assembles the war file for the Java web application and exports it to the Tomcat server. When these tasks are complete the Tomcat server is started. This is seen in Figure 48.

warDeploy.sh

```
#!/bin/bash
$CATALINA_HOME/bin/shutdown.sh
dir=$(pwd)

#Copy gateways
cp -a /home/dan/Docs/BlockchainPrescribing/Orgs/Org1/gateway/. ../Application/app/src/main/resources/
cp -a /home/dan/Docs/BlockchainPrescribing/Orgs/Org2/gateway/. ../Application/app/src/main/resources/

#Copy webapp
cp -r /home/dan/Docs/BlockchainPrescribing/webapp/. ../Application/app/src/main/webapp/

#Remove previous version
cd $CATALINA_HOME/webapps
rm -r app
rm app.war
cd $dir

#Create war file
cd ../Application
gradle assemble
cd app/build/libs/

#Copy new war to directory
cp app.war $CATALINA_HOME/webapps/

$CATALINA_HOME/bin/startup.sh
```

*Figure 47 - warDeploy.sh*

#### 4.5.2.6. deployCC.sh

The functionality of this script is to deploy a Java application as chaincode onto the network. The contract name of the application and the chaincode folder location is passed as command line arguments to the script. The Java application was created using a Gradle-type build system for Java as the chaincode can then be read without error whilst on the network. The script has 5 key parts to it which allow it to deploy chaincode to the network. These parts are discussed with the corresponding code below. The peer command is sourced through the Hyperledger binaries installed as part of the environmental setup.

The first section is compiling the chaincode and then packaging the chaincode, this is the initial functionality which starts the process. This is seen in Figure 48.

deployCC.sh

```
51
52  infoln "Compiling and installing packages for Java code ${C_RESET}"
53  pushd $CC_SRC_PATH
54  ./gradlew installDist
55  popd
56  successln "Finished compiling Java code"
57  CC_SRC_PATH=$CC_SRC_PATH/build/install/$CC_NAME
58
59  . config-env.sh
60
61  packageChaincode() {
62    set -x
63    peer lifecycle chaincode package ${CC_NAME}.tar.gz --path ${CC_SRC_PATH} --lang ${CC_RUNTIME_LANGUAGE}
64    res=$?
65    { set +x; } 2>/dev/null
66    cat log.txt
67    verifyResult $res "Chaincode packaging has failed"
68    successln "Chaincode is packaged"
69  }
```

*Figure 48 - deployCC.sh (Packaging chaincode)*

The second phase is to install the chaincode on each of the peers in the network, this step uses the packaged chaincode created in the first function to install it to the peer as shown in Figure 49.

deployCC.sh

```
71  # installChaincode PEER ORG
72  installChaincode() {
73    ORG=$1
74    setGlobals $ORG
75    set -x
76    peer lifecycle chaincode install ${CC_NAME}.tar.gz >&log.txt
77    res=$?
78    { set +x; } 2>/dev/null
79    cat log.txt
80    verifyResult $res "Chaincode installation on peer0.org${ORG} has failed"
81    successln "Chaincode is installed on peer0.org${ORG}"
82  }
```

*Figure 49 - deployCC.sh (install chaincode)*

The next step is to confirm that the chaincode has been installed successfully on each peer. This is seen in Figure 50.

deployCC.sh

```
84  # queryInstalled PEER ORG
85  queryInstalled() {
86    ORG=$1
87    setGlobals $ORG
88    set -x
89    peer lifecycle chaincode queryinstalled >&log.txt
90    res=$?
91    { set +x; } 2>/dev/null
92    cat log.txt
93    PACKAGE_ID=$(sed -n "/${CC_NAME}_${CC_VERSION}/{s/^Package ID: //; s/, Label:.*$//; p;}" log.txt)
94    verifyResult $res "Query installed on peer0.org${ORG} has failed"
95    successln "Query installed successful on peer0.org${ORG} on channel"
96  }
```

*Figure 50 - deployCC.sh (query installed)*

The next step is to approve the chaincode installed on each peer to the organisation. It is important that both peers approve the same chaincode as the chaincode must be valid from both peer organisations. This is seen in Figure 51.

deployCC.sh

```
98   # approveForMyOrg VERSION PEER ORG
99   approveForMyOrg() {
100    ORG=$1
101    setGlobals $ORG
102    set -x
103    peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example
104    res=$?
105    { set +x; } 2>/dev/null
106    cat log.txt
107    verifyResult $res "Chaincode definition approved on peer0.org${ORG} on channel '$CHANNEL_NAME' failed"
108    successln "Chaincode definition approved on peer0.org${ORG} on channel '$CHANNEL_NAME'"
109  }
```

*Figure 51 - deployCC.sh (approve for Org)*

The final necessary step in the deployCC.sh script is to commit the chaincode definition. This can only be executed once the chaincode has been approved by all organisations in the network. This is seen in Figure 52.

deployCC.sh

```
142  # commitChaincodeDefinition VERSION PEER ORG (PEER ORG)...
143  commitChaincodeDefinition() {
144    parsePeerConnectionParameters $@
145    res=$?
146    verifyResult $res "Invoke transaction failed on channel '$CHANNEL_NAME' due to uneven number of peer an
147
148    # while 'peer chaincode' command can get the orderer endpoint from the
149    # peer (if join was successful), let's supply it directly as we know
150    # it using the "-o" option
151    set -x
152    peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tl
153    res=$?
154    { set +x; } 2>/dev/null
155    cat log.txt
156    verifyResult $res "Chaincode definition commit failed on peer0.org${ORG} on channel '$CHANNEL_NAME' fai
157    successln "Chaincode definition committed on channel '$CHANNEL_NAME'"
158  }
```

*Figure 52 - deployCC.sh (commit chaincode)*

### 4.5.3. Blockchain Chaincode

The blockchain chaincode or the smart-contract application is the name of the code that is deployed to the network. This code acts as an intermediatory between an application and the network itself, to allow data to be manipulated or inserted on the blockchain and providing a utility to do so. Smart contracts are the business logic that governs assets on the blockchain (87). For a smart contract to be deployed to a blockchain network it must be first packaged, hereafter it can then be installed on each of the peers in the network. Once each installation is complete, each peer must approve the chaincode definition to their own individual organisation and once this is complete it can be committed to the network.

This process of needing multiple approvals ensures that one organisation cannot tamper with the ledger and builds trust on the network. For invoke or execute, a transaction with a smart contract each organisation in the network needs to invoke and execute the smart contract on their peer. When the outputs of all the organisations are consistent from the execution of the smart contract the transaction is then added to the ledger.

In this project two separate smart contracts were developed, a prescription-based contract and a user contract. The prescription contract controls the logic surrounding medical prescriptions on the blockchain, this includes the generic methods of insert, view, and update of prescriptions on the blockchain. The user contract controls the logic surrounding user information who have access to the network with similar methods. Each smart contract was created as Gradle Java application. Gradle was used as the structure of the build out provided by Gradle projects integrates easily with the chaincode structure on the blockchain. Initially the chaincode was developed on a maven styled build tool but this caused many issues with the deployment of the chaincode. The contracts are created using Hyperledger's package for creating contracts in Java.

Two separate smart contracts were necessary for this development as to track both changes to prescription data and user data. It was originally planned that user data would operate as a self-sovereign identity which would not be stored on the blockchain but many issues occurred between Hyperledger Indy and Fabric that caused connectivity issues between the two frameworks so the Indy approach was discontinued.

#### 4.5.3.1. Prescription Contract

The first smart contract developed was the prescription contract. This contracts functionality was to allow prescription data to be modified, viewed, or added to the blockchain ledger. This contract has two core parts, the prescription model which specifies the attributes each prescription has and the contract itself which specifies the methods that interact with prescription data on the ledger.

The prescription model class has 14 attributes which describe details about a contract. These attributes were researched previously and discussed in Section 3.6.1. Each of these attributes have an associated get and set method to retrieve or allocate a value. These attributes can be seen in Figure 53.

Prescription.java

src > main > java > prescriptioncontract > Prescription.java > ...

```java
public class Prescription {
    //Values to be stored on the blockchain
    @Property()
    private String PID;
    @Property()
    private String date;
    @Property()
    private String issuer;
    @Property()
    private String owner;
    @Property()
    private String product;
    @Property()
    private String productID;
    @Property()
    private String productPackage;
    @Property()
    private String quantity;
    @Property()
    private String doseStrength;
    @Property()
    private String doseType;
    @Property()
    private String doseQuantity;
    @Property()
    private String instruction;
    @Property()
    private String comment;
    @Property
    private String status;

    //Getters and setters for values
    public String getPID() {
        return PID;
    }

    public void setPID(String pID) {
        PID = pID;
    }
```

*Figure 53 - Prescription attributes*

Another key feature developed in the prescription class were the serialize and deserialize methods. These methods converted prescription objects to and from Json format which is the format that the prescription asset is stored on the ledger.

The serialize method takes the prescription object and converts it to a Json object from this it converts it to bytes so it can be transmitted onto the ledger. This method is shown in Figure 54.

Prescription.java

src > main > java > prescriptioncontract > Prescription.java > ...

```java
    public static byte[] serialize(Prescription prescription) {
       String jsonStr = new JSONObject(prescription).toString();
       return jsonStr.getBytes(UTF_8);
    }
```

*Figure 54 - Prescription serialize*

The deserialize method completes the opposite of the serialize method as it takes byte data that is converted to a string, it converts the string into a Json object and returns back the prescription object retrieved from the value pairs of the Json object. This method can be seen in Figure 55.


*Figure 55 - Prescription deserialize*

The prescription contract itself uses this prescription class to retrieve and access the prescription assets stored on the ledger efficiently. The prescription contract must implement the Hyperledger contract interface. The interface in addition with the Hyperledger “contract” annotation defines it as a smart contract, the annotation specifies details about the contract including the name of the contract which is needed for the deployment of the contract, the contract information is seen in Figure 56. This contract name should be the same as the root project name as the chaincode deployment will fail, this took some understanding to figure out and caused many issues. The contract name of this smart contract is “pc”.

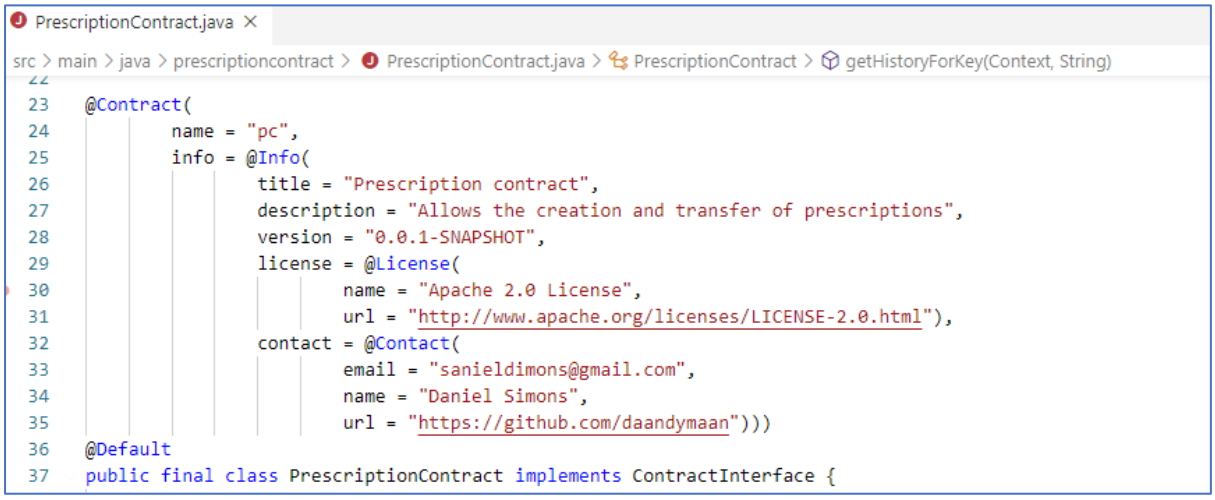



*Figure 56 - Prescription contract information*

The contract is where methods are developed to alter or create prescription data on the blockchain. For each method that had an interaction with the blockchain, the transaction that the method had is specified. This means that the transaction is either adding data or retrieving data from the blockchain, this was defined with “Transaction.TYPE.SUBMIT” or “Transaction.TYPE.EVALUATE” respectively. For each method the context is passed as the first parameter. The context allows for a broad range of ledger API’s including the ability to add and retrieve form the ledger. The context also contains information on the identity that called the function. An example is seen in Figure 57 where the transaction is marked in blue, the context is marked in red and the context being used to retrieve a prescription is in green. These core components are used in every transaction on the contract.

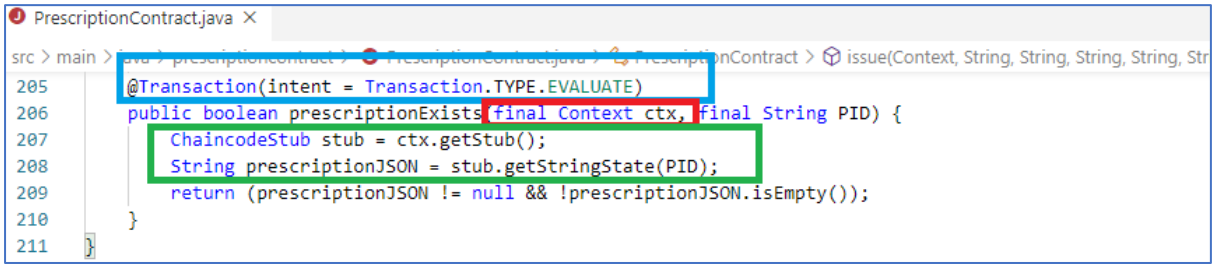



*Figure 57 - Method transaction/context/context use*

In the contract, 8 key methods were developed:

1. generatePID: This method generates a unique PID from the hash values of the date, issuer product and product ID from the prescription. This PID is used to index prescriptions in the ledger and must be unique.
2. Issue: This method creates a prescription from the values passed to it, it uses the generatePID method to generate a PID and setting the status to “ACTIVE” before using the context to add it to the ledger, it then returns the value that was added to the ledger.
3. updatePrescription: This method updates the values of a prescription; it simply creates a new prescription using the values passed to it and then assigns that new prescription to the key of the older version.
4. getAllPrescriptions: This method returns all prescriptions found on the ledger, it does this by iterating through all prescriptions and adding them to an array list. After this it serializes the array list and sends the string response back.
5. transferPrescription: This method is like update prescription except it first retrieves the details of the previous prescription and changes the owner value to be the new value. The method then inserts the new prescription into the ledger and returns the string value of the updated prescription.
6. changeStatus: This method is very similar to the transferPrescription method as it simply updates one field and inserts the prescription back into the ledger.
7. getHistoryForKey: This method displays the benefits of ledged technology as it can retrieve the complete history of an asset on the ledger, meaning it can retrieve the past and present versions of a prescription as no item is deleted from the blockchain. This method returns the complete history of prescription.

8. prescriptionExists: This method is mainly used by other methods in the contract to detect if a prescription asset exists on the ledger It returns a Boolean on the existence of the prescription.

Each of the method returns a string value to confirm the transaction occurred or if an issue occurred. A string was chosen as it can easily be converted to the original object or a Json object. When each of the transactions occur, they first must receive approval by both organisations before they are committed to the ledger.

#### 4.5.3.2. Info Contract

The second smart contract developed was the info contract. The contracts functionality was to keep a record of user's details who have accessed the blockchain. The details can be retrieved and modified using this contract. The contract has two core parts, the user model which specifies the attributes each user and the contract itself which specifies the methods that interact with user data on the ledger.

The prescription model class has 10 attributes which describe details about a contract. The attributes were researched previously and discussed in Section 3.6.1. Each of the attributes have an associated get and set method to retrieve or allocate a value. The attributes can be seen in Figure 58. The cert attribute is the public key associated with the identity of this user.

User.java

src > main > java > infocontract > User.java > User > setTitle(String)

```
7   public class User {
8
9       @Property()
10      private String identifier;
11      @Property()
12      private String title;
13      @Property()
14      private String firstname;
15      @Property()
16      private String surname;
17      @Property()
18      private String address;
19      @Property()
20      private String dob;
21      @Property()
22      private String gender;
23      @Property()
24      private String email;
25      @Property()
26      private String status;
27      @Property()
28      private String cert;
29
30      //Getters and Setters
31      public String getIdentifier() {
32          return identifier;
33      }
34
35      public void setIdentifier(String identifier) {
36          this.identifier = identifier;
37      }
```

*Figure 58 - Info contract attributes*

Like the prescription contract, serialize and deserialize methods were implemented to allow efficient converting of data from the ledger into user objects or converting user objects into data that can be submitted to the ledger. The contract class also implements the same contract interface as the prescription contract. The name of contract is the only difference between the contract annotation definition as seen in figure 59. The contract name is "infocontract".

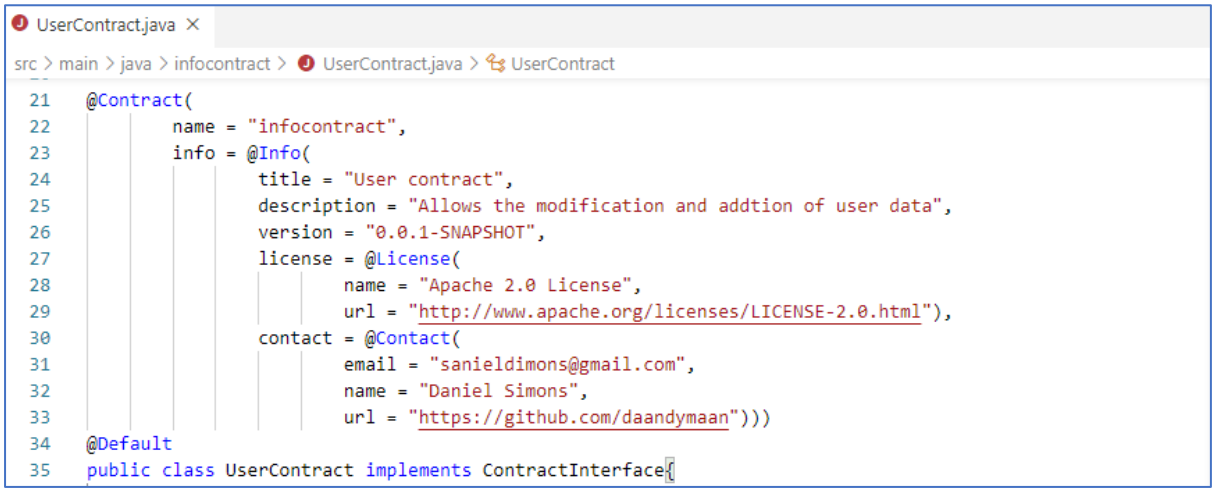



*Figure 59 - info contract definition*

The structure for the methods is similar to the structure of the previous contract. The transaction type must be defined for each method accessing the ledger. The context to gain access to API methods such as adding or retrieving data from the ledger is also used.

There were 4 methods developed in this contract which were:

1. userExists: This method's functionality is to check the existence of user in the ledger. This method is used by many other methods in the contract to check if a user asset exists.
2. createUser: This method's functionality is used to create a new user on the ledger, it is passed all the parameters necessary to create a user and then it is serialized and put onto the ledger.
3. updateUser: The functionality of this method is to update a user based on the parameters passed to it, the method then creates a new user object and assigns it to the identifier onto the ledger.
4. getAllUsers: This method retrieves all users stored on the ledger, it adds each user retrieved to an array list and then serializes the array list and returns the string back.

Each of these method returns a string value to confirm the transaction occurred or if an issue occurred. A string was chosen as it can easily be converted to the original object or a Json object. When each of these transactions occur, they first must receive approval by both organisations before they are committed to the ledger.

For the network to execute these smart contracts, they were installed using the script mentioned in 4.5.2.6. which is the deployCC.sh script. Once these smart contracts were deployed, an application was developed that could use these smart contracts.

### 4.5.4. Conclusion

The data layer consists of a blockchain network where two developed smart contracts for prescriptions and users are deployed. The network allows for identities registered with the network to create, update and view assets on the network. The use of shell scripts allows the automation of many tasks, such as starting and stopping the network and deploying chaincode to the network.

## 4.6. Application Layer Development

This section discusses the application layer development which was the development of the Java web application made using Gradle. This section also discusses the structure and architecture of the web app. The development of this section was based on the research and design discussed in Section 3.5.

### 4.6.1. Application Structure

The application layer of this project was developed in a Java web application using Gradle as the build automation tool. Gradle was used to build a web application resource (WAR) file that could be used to tomcat to host the application on its server. A WAR file is a file used to distribute a collection of jar files, Java classes, Java servlets, webpages and other resources that combine to make a web application. This WAR file can then be read by a tomcat server to host a working and interactive webpage. The WAR file was created in Gradle by using the war plugin for Gradle which compresses the classes, jar files and webapp folder into a WAR file. The WAR file structure is shown in Figure 60. This WAR file was deployed to the Tomcat server using the “warDeploy.sh” script discussed in Section 4.5.2.5.

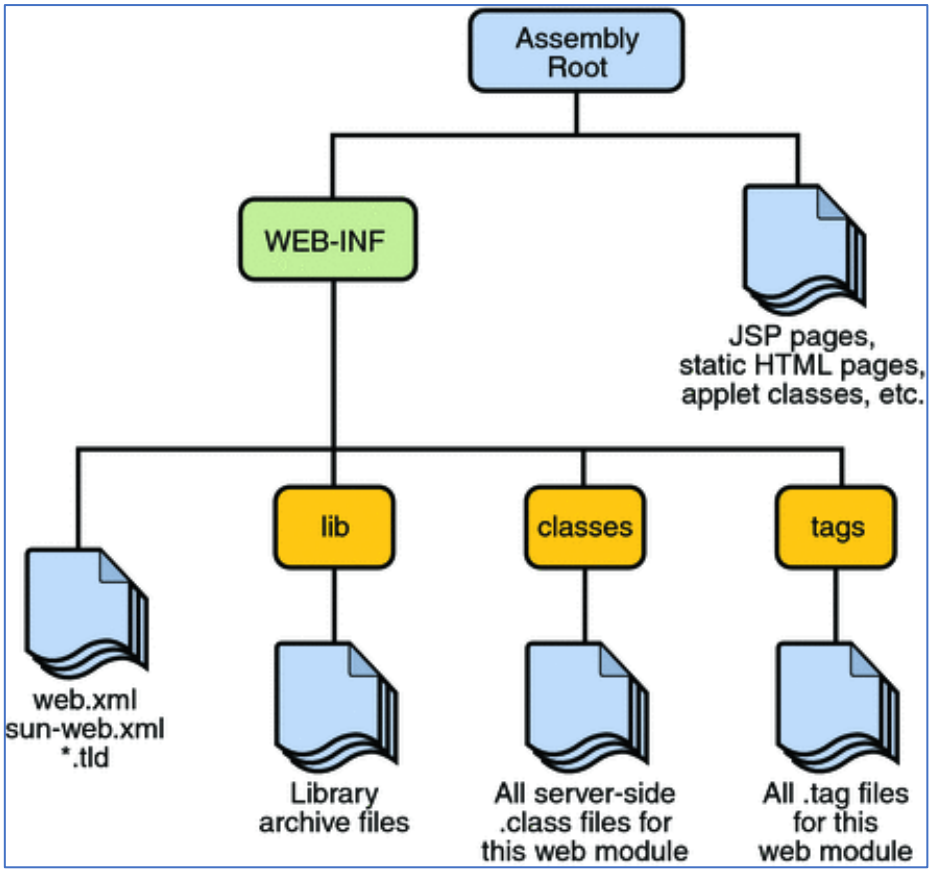



*Figure 60 - WAR file structure*

The application itself was developed with the model view controller (MVC) architectural pattern, which was discussed in the Section 3.5.1. There were three main components of the web application that made the MVC architecture, Json objects, service classes and the request classes. Json objects were used to store data to and from the ledger and were chosen for the model of this application. Only primitive type data could be sent and received from the ledger, this meant that custom data types and objects could not be sent to the ledger. Json objects were selected as a standardized way to store data that was sent to and from the ledger as it allowed data to be easily converted to string format and could easily be parsed from String requests sent from the presentation layer examples of this are shown in Figure 61.

UserRequests.java

app > src > main > java > application > requests > UserRequests.java > UserRequests > userExistsByCert(JsonObject, String)

```
46      public static String getUserByIdentifier(JsonObject user, String identifier) {
47          JsonObject userSelected = new JsonObject();
48          JsonArray JSONResults = JsonParser.parseString(UserRequests.getUsers(user)).getAsJsonArray();
49          LOGGER.info(JSONResults.toString());
50          for (JsonElement jsonElement : JSONResults) {
51              if (jsonElement.getAsJsonObject().get("identifier").getAsString().equals(identifier)){
52                  userSelected = jsonElement.getAsJsonObject();
53              }
54          }
```

*Figure 61 - Json example in Java*

The Json objects themselves were modelled of the entity classes “Prescription” and “User” defined on the chaincode. The entity classes acted as blueprints for the attributes of the assets that were stored on the ledger although even still only Json data was stored on the ledger. The “Prescription” entity and the “User” entity are seen in Figures 55 and 60 respectively.

The view portion of the architecture was the classes stored in the service directory of the application. These classes contain URI path’s which allow HTTP requests to be sent from the user. Each method contains a URI path to access a different resource an example is seen in Figure 62.

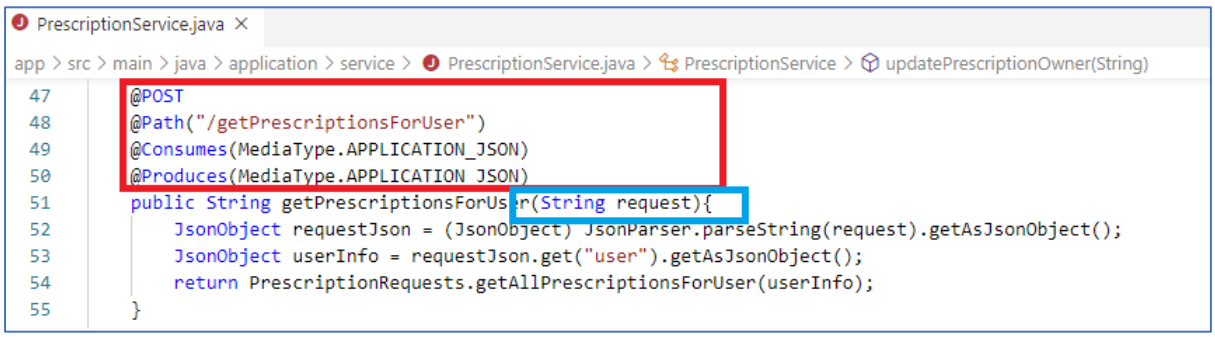



*Figure 62 - View component Service class*

This URI paths were created using JAX-RS which was discussed in Section 3.5.3. Each of the methods receives a request from the user and the returns the response back. Each request and response are Json string values which allow the values to be easily converted to and from Json.

The controller component of the MVC architecture was represented in the requests directory. The request classes managed and altered data on the ledger. It provides a logical link between the service classes and the ledger or network itself. The network or ledger is accessed with the Connection class which gives the controller classes access to this network. The appropriate smart contract is called depending on what the user requests and data is inserted or received from the ledger. Each method in controller class receives the appropriate data for the method and each method accesses the ledger and returns the result as byte data. The data is then converted to a String before returning it to the service method.  An authorized user’s details are needed for each request to the network which is why the Json object is passed. The user is then used to gain access to the network and the appropriate contract is selected. When the contract is selected, either “submitTransaction” or “evaluateTransaction” methods are used to access the smart contract on the blockchain. The parameter passed is equivalent to the method name on the smart contract. This is seen in Figure 63.

PrescriptionRequests.java

app > src > main > java > application > requests > PrescriptionRequests.java > PrescriptionRequests > transferPrescription(JsonObject, Strir

```
56      public static String getAllPrescriptions(JsonObject user) {
57          byte[] result;
58          try {
59              Contract contract = Connection.getContract(user, contractName);
60              result = contract.evaluateTransaction("getAllPrescriptions");
61              return new String(result);
62          } catch (Exception e) {
63              e.printStackTrace();
64              JsonObject errorResponse = new JsonObject();
65              errorResponse.addProperty("msg", "Prescriptions could not be found");
66              return errorResponse.toString();
67          }
68      }
```

*Figure 63 - Controller request class*

Both the controller classes and the view classes are separated into their own folders within the application to isolate the components and give clear separation of concerns within the application.

Each group of classes with similar functionality were organised into folders within the Java application. The controller classes were separated into the requests folder while view classes were sorted into the service folder. Other classes were also grouped together into folders such as classes that aided in connecting to the network like “Connection.java”, “Authentication.java” and “RegisterUser.java” which were categorized into the “util” folder. The singleton logger class “Logging.java” was also put in a separate folder. The session functionality with the Tomcat server was put into its own folder. These were classes which aided in the creation, retrieval, and termination of a HTTP session. The complete folder structure is seen in Figure 64.

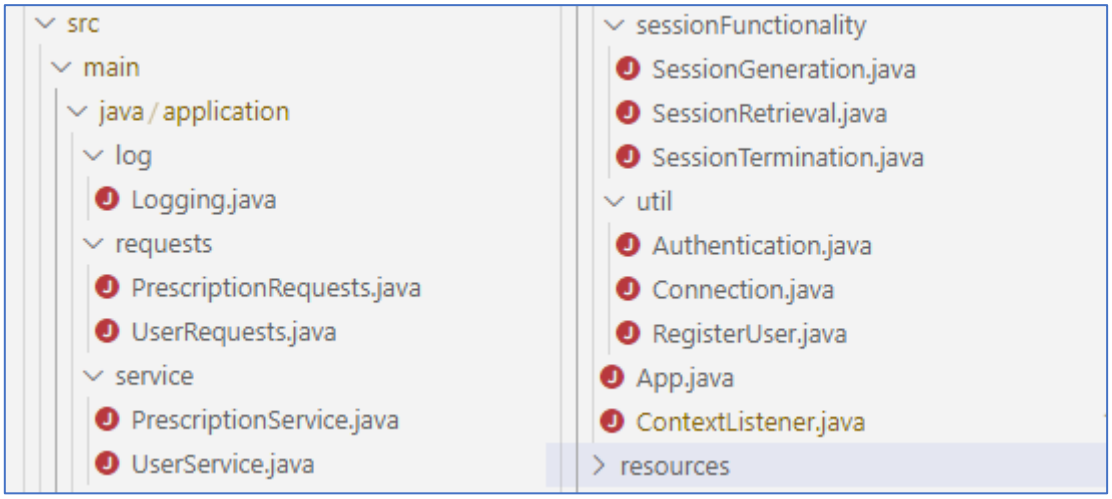



*Figure 64 - Application structure*

### 4.6.2. RESTful architecture

As discussed in Section 3.5.2, a RESTful architecture was used to allow stateless transfer of data between the user and the application itself. The architecture was developed in Java using a combination of JAX-RS, a package which enables simple RESTful architecture within

Java and Tomcat a server to which the application was hosted on. Jersey was also used with JAX-RS which allowed automatic specification of resource URI formats.

As discussed in Section 4.6.1, JAX-RS, URI annotations were implemented in the service classes of the project. The classes acted as gateways for the client to access resources from the application which in turn accessed resources from the ledger itself. An application path was first implemented which is the base URI for the application this is shown in figure 65. This is necessary for resources from that application to be accessed.

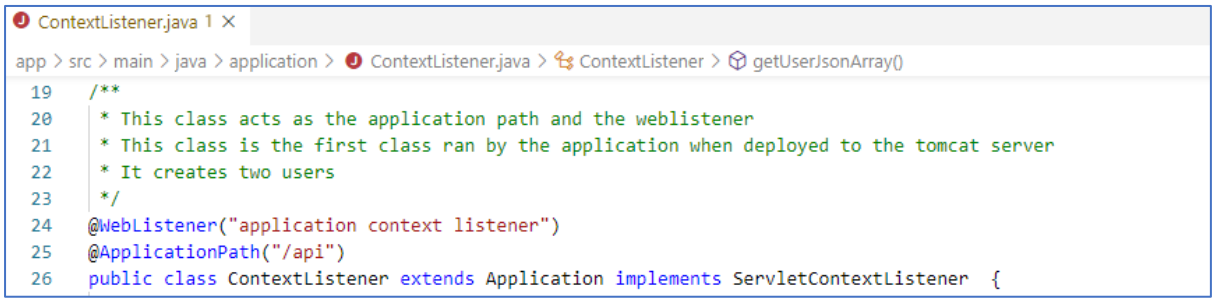



*Figure 65 - Application path (api)*

Once the application path was specified the URI paths for each resource were specified. There was a total of two classes which contained methods for accessing resources from the blockchain. These were the two service classes “PrescriptionService.java” and “UserService.java” located in the service folder of the application. A URI path for each class was defined for the resources to be accessed from that specific class this is show in Figure 66.

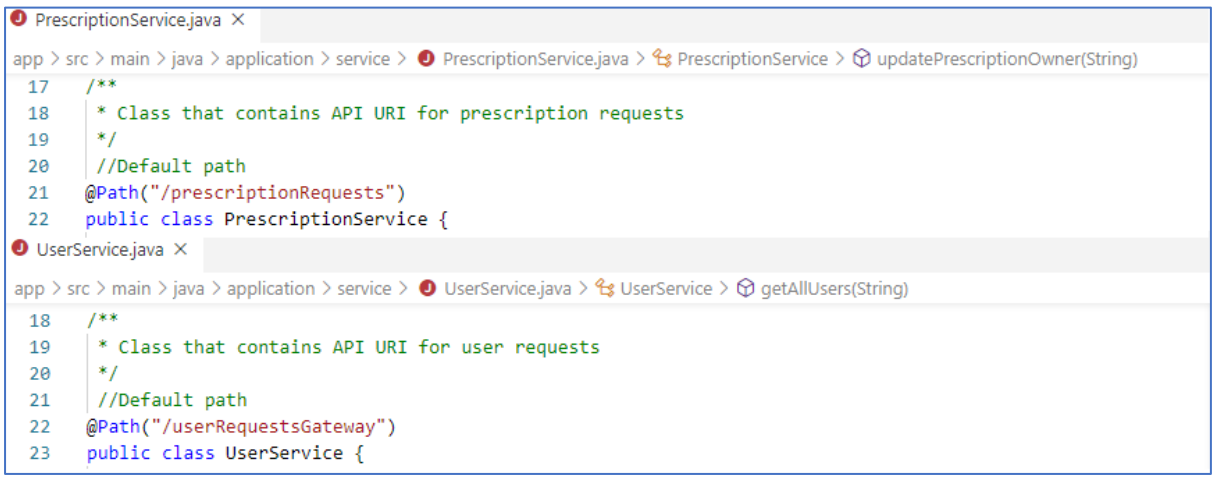



*Figure 66 - Class URI paths*

As mentioned previously each method in each of these classes is assigned a URI path so that the client can access resources from the application. Requests made to the application are HTTP requests, there a various forms of different HTTP request such as GET, POST, PUT and DELETE to name a few. For this application HTTP POST requests were chosen for each method as the data is stored in the body of the request which allows more data to be sent. The annotation for the HTTP request method is the first annotation found above each of the resource methods. After the HTTP request method is defined, the URI path for that resource

is defined so that the specific resource is accessed. Important aspects of creating the resource method is to use annotations to define what the data format sent to the method is and the data format returned from the method. For this application Json format for data was selected as Json data is easily parsed to and from a String, is recognizable and easily manipulated in JavaScript and is easily to and from the blockchain ledger. Json format is used throughout the entire program and proved the easiest format to integrate with it. To define the data format that is sent in the request and returned in the response the annotations "@Consumes" and "@Produces" were used. The full set of these annotations is seen in Figure 67 which is the method "updatePrescriptionOwner" which is the method that allows access to changing the ownership of a prescription such as transferring a prescription.

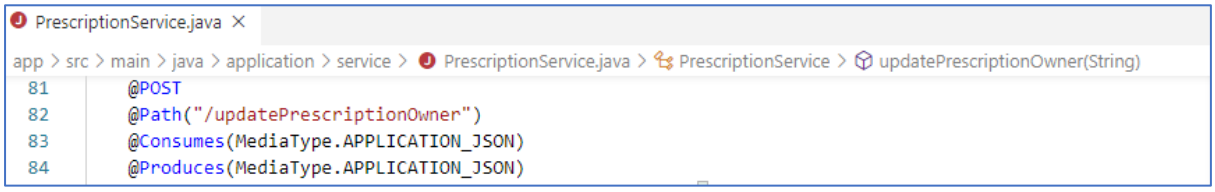



*Figure 67 - JAX-RS annotations for method*

Although the HTTP request contains content that is in Json format, Json format is not one of Java's core datatypes. To use Json in Java a package or library was added to the build of the Java project, in this application the Json library provided by Google was used. This Json cannot be used however to receive the data directly from the request and instead this data must be converted to a Json object as Json is not a primitive datatype of Java. Fortunately, this data is received as a String and converted easily to Json format using "JsonParser", a class found in the Json package provided by Google which makes converting Json Strings into Json Objects very easy. An example of this is shown again in the "updatePrescriptionOwner" class shown in Figure 68, but this conversion is found in each method from the service classes.

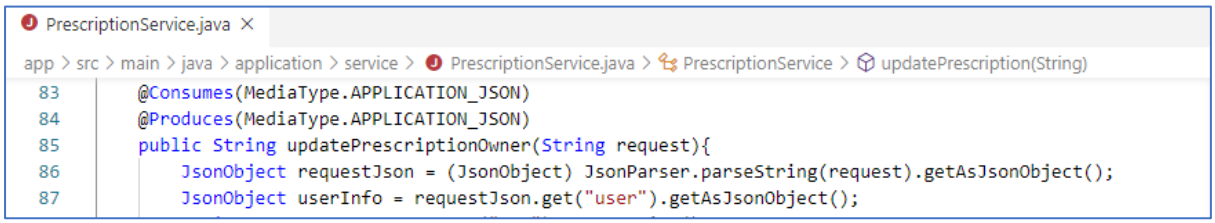



*Figure 68 - Converting request data to Json*

Once the data is converted into a Json object the relevant fields from the Json object are extracted and used by the controller classes. The controller classes return data manipulates and uses the Json Object but converts the Json object back to a String so that the appropriate response is sent back. The full "updatePrescriptionOwner" method is shown in Figure 69.

PrescriptionService.java

app > src > main > java > application > service > PrescriptionService.java > PrescriptionService > updatePrescription(String)

```
75      /**
76       * POST request
77       * Updates owner of prescription (Transfer)
78       * @param String request(JsonObject)
79       * @return MediaType.APPLICATION_JSON (JsonObject.toString())
80       */
81      @POST
82      @Path("/updatePrescriptionOwner")
83      @Consumes(MediaType.APPLICATION_JSON)
84      @Produces(MediaType.APPLICATION_JSON)
85      public String updatePrescriptionOwner(String request){
86          JsonObject requestJson = (JsonObject) JsonParser.parseString(request).getAsJsonObject();
87          JsonObject userInfo = requestJson.get("user").getAsJsonObject();
88          String PID = requestJson.get("pID").getAsString();
89          String owner = requestJson.get("owner").getAsString();
90          String newOwner = requestJson.get("recipient").getAsString();
91          return PrescriptionRequests.transferPrescription(userInfo, PID, owner, newOwner);
92      }
```

*Figure 69 - updatePrescriptionOwner (Transfer prescription) method*

For the “PrescriptionService.java” file, 7 URI paths were implemented these were:

1. “/getAllPrescriptions”: Gets all prescription entities on the ledger
2. "/getPrescriptionsForUser": Gets all prescriptions for a user from the ledger
3. "/updatePrescriptionStatus": Updates prescription status
4. "/updatePrescriptionOwner": Updates owner of prescription (Transfer)
5. “/updatePrescription": Updates the prescription (All details)
6. "/createPrescription": Creates a new prescription on the ledger
7. "/getPrescriptionHistory": Gets full prescription history of a prescription on the ledger

For the “UserService.java” file, 5 URI paths were implemented these were:

1. "/getAllUsers": Gets all users stored on the ledger
2. "/getUser": Gets a user's detail by identifier
3. "/updateUserDetails": Updates a user's details
4. "/createUser": Creates a user by HTTP request
5. "/userExists": Detects if a user exists by wallet address

3 separate URI paths exist that are not in either of the service classes, these are URI paths which are for the session of a user. This requires session information to be sent as the request and not just data. These are:

1. “/userRequestsGateway/authenticateUser”: Authenticates user and creates a new session if the user’s details are correct
2. "/userRequestsGateway/verifySession": Retrieves the session information from the user, which includes the identifier, status and cert associated with them.
3. “/userRequestsGateway/logout”: Terminates a session and effectively logs out the user from the webpage.

### 4.6.3. Registering with the ledger

To use the ledger within the application, an identity must be used. An identity is a set of Json data which holds the certificates and keys needed to access to specified network. An example of an identity is seen in Figure 70. This identity uses the X.509 certificate and defines which organisation it belongs to. As Hyperledger Fabric is a private blockchain, only users with a registered identity can access the data within it, limiting the network to authorized users and thus securing the data within the blockchain.

```
{
  "version": 1,
  "mspId": "Org1MSP",
  "type": "X.509",
  "credentials": {
    "certificate": "-----BEGIN CERTIFICATE-----
                    MIIChTCCAiygAwIBAgIUEnPeKM7VGYvJmt7GqSkjs7EVJFowCgYIKoZIzj0EAwIw
                    cDELMAkGA1UEBhMCVVMxFzAVBgNVBAgTDk5vcnRoIENhcm9saW5hMQ8wDQYDVQQH
                    EwZEdXJoYW0xGTAXBgNVBAoTEG9yZzEuZXhhbXBsZS5jb20xHDAaBgNVBAMTE2Nh
                    Lm9yZzEuZXhhbXBsZS5jb20wHhcNMjEwMzI2MTM0NDAwWhcNMjIwMzI2MTM0OTAw
                    WjBFMTAwDQYDVQQLEwZjbGllbnQwCwYDVQQLEwRvcmcxMBIGA1UECxMLZGVwYXJ0
                    bWVudDExETAPBgNVBAMTCDMzMjQ3ODRWMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcD
                    QgAEjyTzRy3mX+GMSwxgS6C3fh+OY2Kfl68ArxLySrMS51C9Z9kqFkRROFXLlf0Z
                    M8iqKCfpKSDWp+7XNC7K+fbVNKOBzjCByzAOBgNVHQ8BAf8EBAMCB4AwDAYDVR0T
                    AQH/BAIwADAdBgNVHQ4EFgQUMpFfgslP2C+PNKdrswJ70Zri3m8wHwYDVR0jBBgw
                    FoAUSlMpqbRtfJrels1rD07keH64Tv8wawYIKgMEBQYHCAEEX3siYXR0cnMiOnsi
                    aGYuQWZmaWxpYXRpb24iOiJvcmcxLmRlcGFydG1lbnQxIiwiaGYuRW5yb2xsbWVu
                    dElEIjoiMzMyNDc4NFYiLCJoZi5UeXBlIjoiY2xpZW50In19MAoGCCqGSM49BAMC
                    A0cAMEQCIDLuZg3yq19XMBR3cy4yNJOz5WgJIiKl2WI64i8wAK6fAiBoE1D4B0pG
                    nbJ68i70Ba0dMAKE55p/QKL1oay5kSdh+YA==
                    -----END CERTIFICATE-----",
    "privateKey": "-----BEGIN PRIVATE KEY-----
                   MIGTAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBHkwdwIBAQQg1z31xwXjhDsxRGc9
                   Ma7RCFzJFQ5jDJ606jEzgWNtdaigCgYIKoZIzj0DAQehRANCAASPJPNHLeZf4YxL
                   DGBLoLd+H45jYp+XrwCvEvJKsxLnUL1n2SoWRFE4VcuV/RkzyKooJ+kpINan7tc0
                   Lsr59tU0
                   -----END PRIVATE KEY-----"
  }
}
```


*Figure 70 - Example of an identity*

In the application the class “RegisterUser.java” in the util folder was used to generate identities for the network. To create a new identity on the network, an admin identity is needed. New identities can only be approved by an admin who is already connected to the network, this protects the security of the network as only approved users have accessed to the ledger. The other component needed to connect a new identity to the network is the certificate authority of the network. As mentioned previously in Section 4.5.1, the certificate authority controls the access rights of identities to the network. With these two components an X.509 identity can be created. The identities are saved as files to a directory called the “wallet” directory. The identities saved were named after the unique identifier associated with the identity such as a PPS number or healthcare identifier. After this identity is created a user asset with the appropriate attributes is added to the ledger to store the details about the new user that was created. Figure 71 shows the code implemented used to create a new identity and then register the user with the ledger.

RegisterUser.java

app > src > main > java > application > util > RegisterUser.java > RegisterUser > getCert(JsonObject, Wallet)

```
93
94          //Creating admin X509 identity
95          X509Identity adminIdentity = (X509Identity)wallet.get("admin");
96          User admin = getAdminUser(adminIdentity);
97
98          //Enrollment request on the network using the CA
99          RegistrationRequest registrationRequest = new RegistrationRequest(user.get("identifier").getA
100         registrationRequest.setAffiliation("org1.department1");
101         registrationRequest.setEnrollmentID(user.get("identifier").getAsString());
102         String enrollmentSecret = ca.register(registrationRequest, admin);
103
104         //Creating identity with enrollment request
105         Enrollment enrollment = ca.enroll(user.get("identifier").getAsString(), enrollmentSecret);
106         Identity newUser = Identities.newX509Identity("Org1MSP", enrollment);
107         wallet.put(user.get("identifier").getAsString(), newUser);
108         LOGGER.info("Succesfully enrolled " + user + " and imported into wallet");
109
110         //Creating user information on the network.
111         user = getCert(user, wallet);
112         String response = UserRequests.createUser(user, user.get("identifier").getAsString(), user.ge
113         LOGGER.info("Created user entry on the ledger" + response);
```

*Figure 71 - RegisterUser.java (Creating a new identity)*

### 4.6.4. Connection to ledger

The connection to the blockchain ledger was also a key aspect of the development of the web application. The connection to the ledger allowed data to be transferred and retrieved using the smart contracts discussed earlier in section 4.5.3. The “Connection.java” class found in the util folder of the application was created to allow an easy and straight forward connection to the ledger which allowed access to the methods of the smart contract. This connection class file was made up of three methods; connect, getWallet and getContract.

As discussed in Section 4.6.3, an identity that is registered with the network is needed to access the ledger, these identities are stored in the directory labelled “wallet”.

The first step of connecting to the ledger is to retrieve the identity of the user attempting to access the ledger. This is done by retrieving the identity associated with the users attribute of “identifier”.

Once the identity of the user is found, a connection to the network can be made, this connection is found using the connection information stored in the “Orgs” folder discussed in section 4.3. From this network connection, the contract needed can be returned. This process is shown in Figure 72.

Connection.java

app > src > main > java > application > util > Connection.java > ...

```
25      * Gets Hyperledger network connection to the blockchain using a user's credentials
26      *
27      * @param JsonObject user
28      * @return Network network
29      * @throws Exception
30      */
31     public static Network connect(JsonObject user) throws Exception{
32         Wallet wallet = getWallet();
33         Path networkConfigPath = Paths.get(connectionDIR);
34         Gateway.Builder builder = Gateway.createBuilder();
35         builder.identity(wallet, user.get("identifier").getAsString()).networkConfig(networkConfigPath).d
36         Gateway gateway = builder.connect();
37         Network network = gateway.getNetwork("mychannel");
38         return network;
39     }
40
41     /**
42      * Retrieves a specified contract from the network.
43      * @param JsonObject user
44      * @param String contractName
45      * @return Contract contract
46      * @throws Exception
47      */
48     public static Contract getContract(JsonObject user, String contractName) throws Exception {
49         Network network = connect(user);
50         Contract contract = network.getContract(contractName);
51         return contract;
52     }
53
54     /**
55      * Retrieves the wallet with credentials
56      * @return Wallet wallet
57      * @throws Exception
58      */
59     public static Wallet getWallet() throws Exception{
60         Path walletpath = Paths.get(walletDIR);
61         Wallet wallet = Wallets.newFileSystemWallet(walletpath);
62         return wallet;
63     }
```

*Figure 72 - Connection.java*

The contract can then be used to access methods from the smart contract deployed on the blockchain network. There two methods used by this application from the contract object are the “evaluateTransaction” and “submitTransaction”. The methods, depending on what transaction type was assigned to the method on the smart contract, are used to execute transactions on the network using the smart contract. The method is passed the smart contract method name and the parameters needed for the method. The data returned from the transaction is stored into a byte array as the data transferred across the network is not stored in any specific data type. This byte data can easily be converted into a String and then from the String converted into Json data. The process is repeated throughout the two controller classes found in the requests folder of the application an example of one of the methods from the “PrescriptionRequests.java” class is seen in Figure 73.

PrescriptionRequests.java

app > src > main > java > application > requests > PrescriptionRequests.java > PrescriptionRequests > getAllPrescriptionsForUser(JsonObje

```
93      /**
94       * This method transfers ownership of a prescription to another user
95       * @param JsonObject user
96       * @param String PID
97       * @param String owner
98       * @param String newOwner
99       * @return String
100      */
101     public static String transferPrescription(JsonObject user, String PID, String owner, String newOwner)
102         byte[] result;
103         JsonObject errorResponse = new JsonObject();
104         try {
105             if(getPrescriptionByPID(user, PID) == null){
106                 errorResponse.addProperty("msg", "Prescription does not exist");
107                 return errorResponse.toString();
108             }
109
110             JsonObject prescriptionChosen = JsonParser.parseString(getPrescriptionByPID(user, PID)).getAs
111             if(!prescriptionChosen.get("owner").getAsString().equals(owner)){
112                 errorResponse.addProperty("msg", "This prescription is owner by another user");
113                 return errorResponse.toString();
114             }
115
116             if(UserRequests.userExistsByCert(user, newOwner)){
117                 Contract contract = Connection.getContract(user, contractName);
118                 result = contract.submitTransaction("transferPrescription", PID, owner, newOwner);
119                 return new String(result);
120             } else {
121                 errorResponse.addProperty("msg", "Recipient address incorrect: User does not exist");
122                 return errorResponse.toString();
123             }
124         } catch (Exception e) {
125             e.printStackTrace();
126             errorResponse.addProperty("msg", "Prescriptions could not be transfered");
127             return errorResponse.toString();
128         }
129     }
```

*Figure 73 - transferPrescription method*

### 4.6.5. Authentication with the ledger

Authentication is a vital process to this application as it is important that access to the ledger is only permitted to user's who have been granted access to the ledger. The main components of authentication are the user's identifier such as their PPS number and the last 8 characters of their identities private key. The private key is only disclosed to the owner of the key and the identifier is a unique identifier guaranteeing that the user accessing the network is the user that was registered with the network. Only 3 of the characters are requested by the application in a random order. The identifier, passcode pattern and the passcode itself is sent to the application for authentication.

The authentication process is simply comparing the characters entered by the user to the characters that appear in the private key of the users identity. The first step in the process is to verify that user's identifier or identity exists on the ledger by checking to see if an identity exists with the user's identifier. After the identity is found, the next step is to verify that the passcode that the user entered is correct. As the passcode entered is 3 random digits, the passcode pattern is also needed. The index of the passcode is matched to the correct

passcode found in the identity. This method is the “verifyPassphrase” and is shown in Figure 74. If the details that the user entered are correct the next step is started but if they are incorrect the authentication is halted until new details are entered by the user.

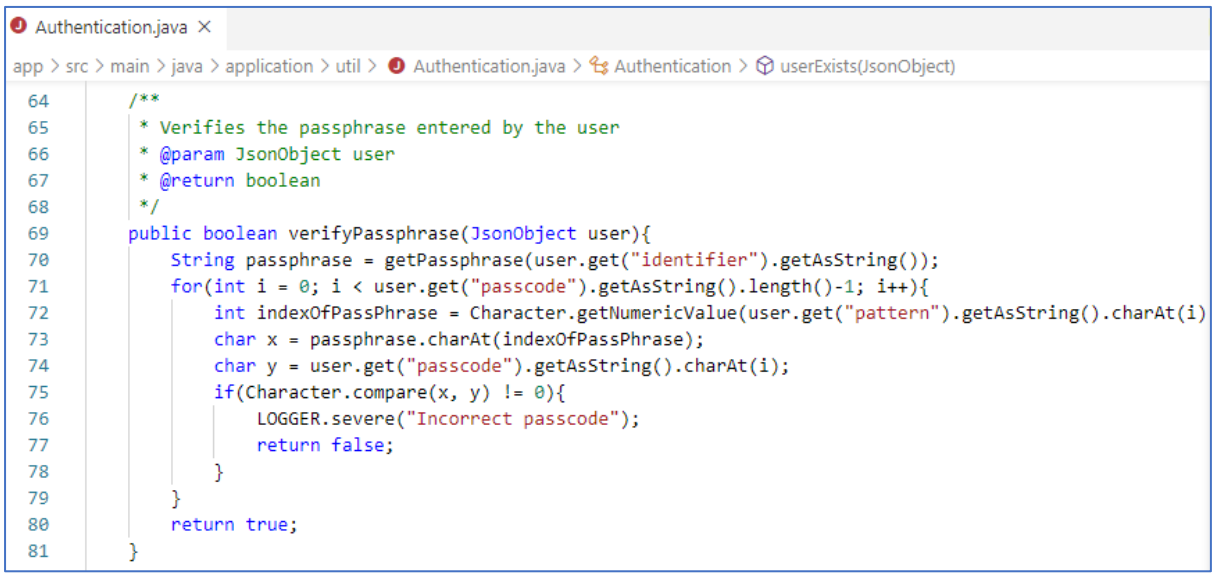



*Figure 74 – verifyPassphrase*

The next step in the authentication process is to retrieve the data associated with the user. The data is the certificate and the status associated with the user and is used for various functionality within the application.

One of the functions of this data is to store the data onto the session associated with the user. HTTP sessions are used to monitor a user’s access to a web application by storing the details associated with the user onto the server. This way a user can be verified to be the same user even if the user changes webpage. Sessions are created using the HTTP request sent to the method. If there is no session a new session is created. This is shown in Figure 75.


*Figure 75 - Session generation*

The attributes associated with the user are then added to the session using the session.addAttribute method. The session values can then be accessed through the HTTP requests sent from the client. In Figure 76, values are assigned to the session variables.


*Figure 76 - Session variables*

With session variables now stored on the session, requests made by the user are verified so that access is granted to the specific user who logged in. Once the user logs out of the webpage the session is terminated to ensure no other users attempt to use access an account that is not theirs. The session termination is shown in Figure 77.


*Figure 77 - Session termination*

### 4.6.6. Conclusion

This section discussed the development process of the application layer of this project. The structure of the WAR file and how it was deployed to the Tomcat server. The details of the MVC structural architecture of the application and how each component worked with the other. The RESTful architecture of the application which allowed the stateless access of resources from the presentation layer using URI paths. The process of registering a user to access the blockchain network was also discussed. Finally, the process involved in the authentication of a user on the network and the HTTP session associated with that user.

## 4.7. Presentation Layer Development

This section discusses the development of the presentation layer. This includes the implementation of the HTML pages that make the webpage and the JavaScript code that creates the interactive and functionality of the webpage. The presentation layer was developed in accordance with the research and the design planned in Section 3.4.

### 4.7.1. Presentation Layer Structure

The presentation layer is made up of HTML, CSS, and JavaScript files. The HTML and CSS files are used for creating the static webpages while the JavaScript gives the webpages interactive and functionality. The website was developed in the webapp folder and then

copied to the webapp folder located in the web application. The process was completed using the “warDeploy.sh” script discussed in Section 4.5.2.5.

The folder structure of the webapp was created to separate out the different types and groups of files together. Each webpage has its own folder to separate the webpages from each other. The folders contain the HTML file associated with that page the only exception to this is the “index.html” page which is in no directory.

Each HTML file implemented the structure of each webpage, this includes navigation bar. Some elements of the HTML were implemented using JavaScript such as populating tables or creating widgets which contained data from the ledger. The style of the webpage was created mainly using bootstrap 4. This was to give a uniform design to the webpage that also gave style to it. Bootstrap also included some JavaScript code which gave functionality to some of the HTML elements implemented into the webpage.

There is one CSS file, “style.css” which controls the CSS of all the HTML files within the webapp directory.

Images needed by the each of the HTML files in this directory are stored in the “images” folder.

JavaScript files which control the functionality of the webpage, are stored in the “utils” folder. The most important JavaScript file within the util folder is the “Server.js” file. This file makes the HTTP request to the application layer of the project. These requests are made using an Ajax call which is from the Jquery library. These requests are formatted in JSON and each separate resource request has its own method in the “Server.js” file. Each method contains a specified request URL that relates to the specific resource the method wants to access from the application layer. Each request contains the user’s details so that a connection to the ledger can be made.

Finally, the WEB-INF folder contains the web.xml files which controls the settings associated with the webpage.

The complete structure of the webapp directory is shown in Figure 78.

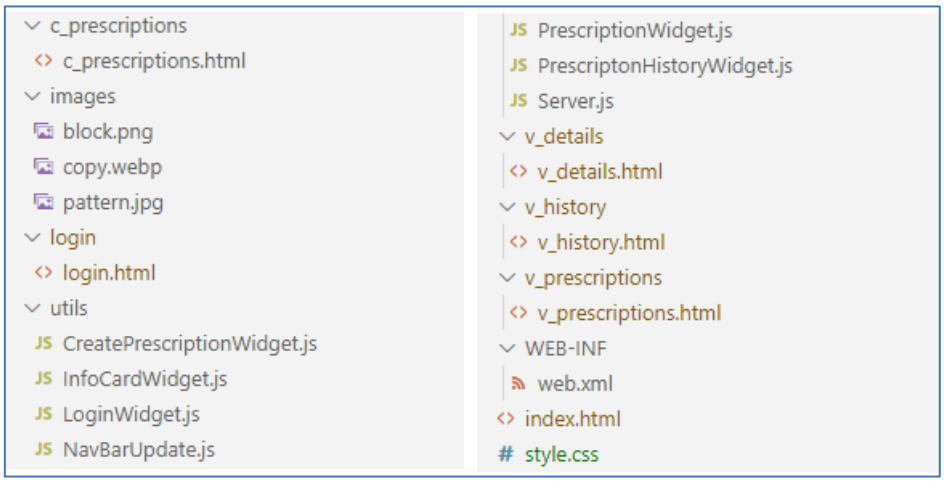



*Figure 78 - Webapp folder structure*

### 4.7.2. User Authentication

To access the webpage a user must first log in. The webpage will redirect the user to the login page if they attempt to access any part of the webpage not logged in with an account. This was implemented in the "NavBarUpdate.js" file which checks to see if the user is currently in a HTTP session and if they are not it redirects the user to the login page which is seen in Figure 79.

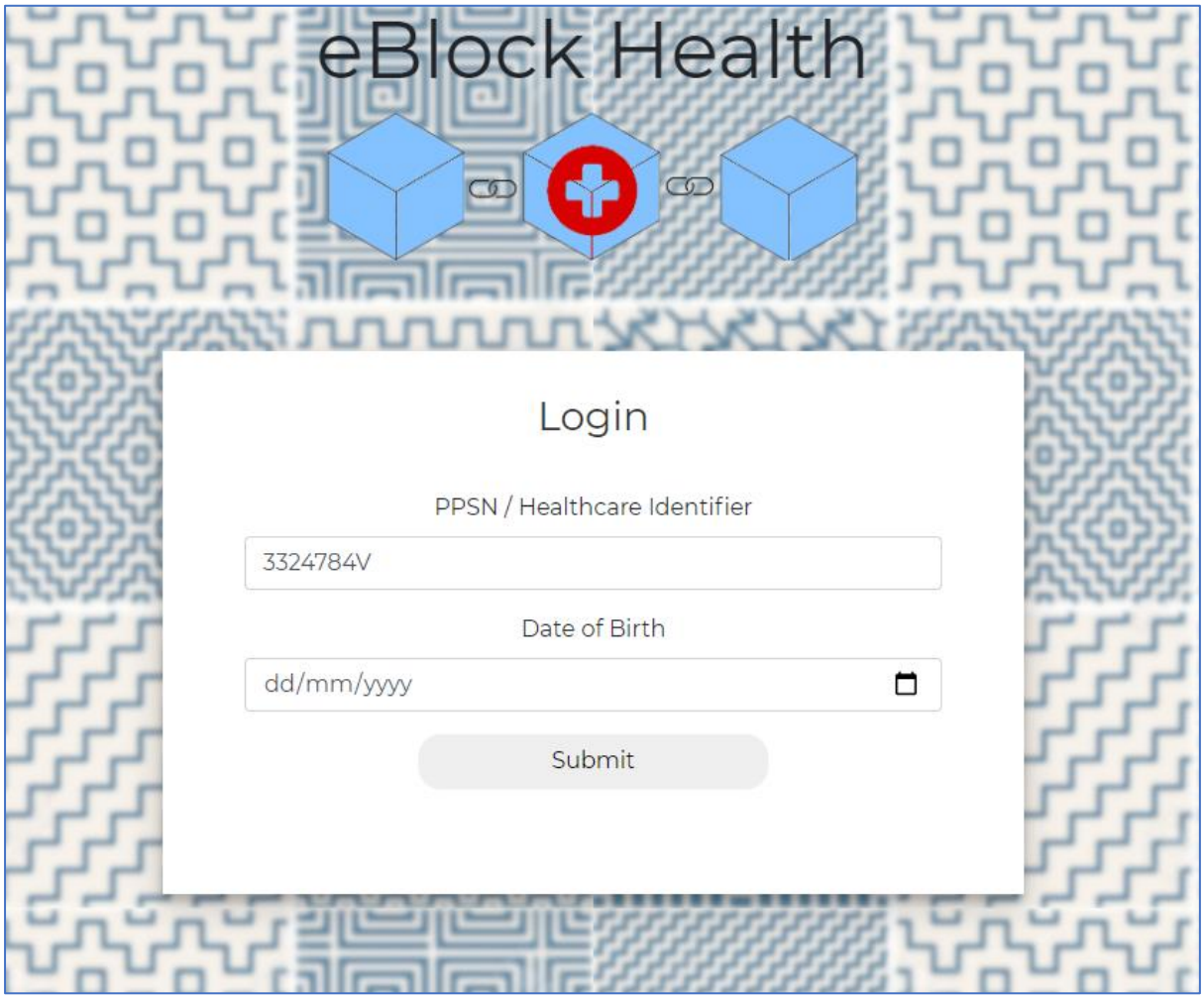



*Figure 79 - Login screen*

The login page consists of 3 fields, which are the identifier, date of birth and then the passcode. The login page first requests the identifier and the date of birth of the user. The identifier and date of birth are both sent to the application layer using an ajax method call found in the "Server.js" file. The "Server.js" file contains all HTTP requests to the application layer, it uses the Jquery implementation for an ajax request which is seen in Figure 80. Each method in the "Server.js" file contains a specific URL for the resource it is trying to access. It sends this URL along with the data to be sent to the application layer to the ajax method which then it turns sends it to the requested URI path in the application layer.

Server.js

utils > Server.js > createPrescription > verifySession() callback

```
function postRequest(url, data, callback){
    console.log("URL:", url);
    console.log("request:", data);
    $.ajax(url, {
        dataType: 'json',
        // dataType: 'text',
        method: 'POST',
        contentType: "application/json; charset=utf-8",
        data: JSON.stringify(data),
        success: function (data, status, xhr) {
            console.log("response:", data);
            if (callback) callback(data);
        },
        error: function (jqXhr, textStatus, errorMessage) {
            console.log(errorMessage);
        }

    })
}
```

*Figure 80 - Server.js (Ajax post request)*

The request is sent to check if a user exists with the details entered by the user, if a user exists with the details entered by the user, a passcode dialog appears. The passcode dialog asks that the user enter 3 of the 8 characters of their private key passphrase. The 3 characters needed by the login page are randomly selected. The pattern that was selected and the passphrase entered by the user is sent to be verified along with the identifier the passphrase is for. The authentication request is sent again with an ajax post method where the data is sent in JSON format. This request is sent using a specified REST API call. These REST calls were discussed in Section 4.6.2. An example of what the passcode dialog box is shown in Figure 81.

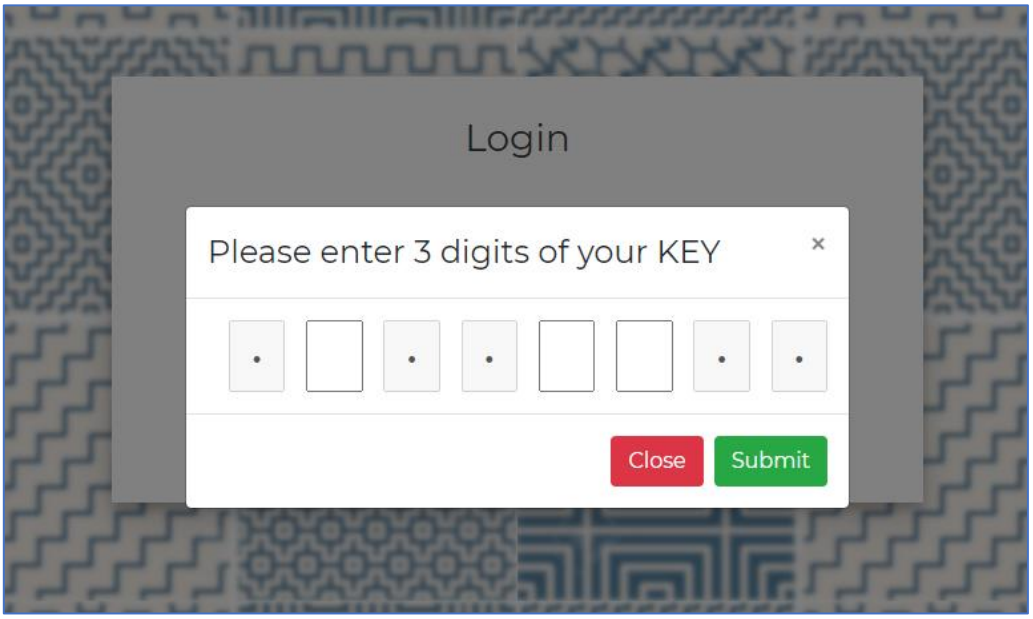



*Figure 81 - Passcode enter screen*

Once a user has logged in, the “NavBarUpdate.js” file returns the details from the current HTTP session. These details are the user’s identifier, status and certificate that were attributes saved to the session from the authentication process which was discussed in Section 4.6.5. These attributes are used to display the appropriate information to the user and to request information from the application using the users details. These attributes aid in displaying the correct identifier on the right-hand side of the navigation bar of the website. This link of the navigation bar has a drop-down feature to allow the user to select from two options, to view their personal details stored on the ledger and to log out of their account on the website. This feature of the navigation bar is shown in Figure 82.


*Figure 82 - Navigation bar update*

The logout link Is used to terminate the user’s HTTP session and bring the user back to the login page. The logout link sends a HTTP post request to the application which in turn terminates the HTTP session associated with it. This session termination is briefly discussed in Section 4.6.5.

### 4.7.3. User Information

As mentioned in Section 4.7.2, the “NavBarUpdate.js” file updates the navigation bar to include to extra links to view the user’s details and log out of the website. The “My details” link found in the drop-down items brings the user to a webpage which displays the user’s details which are stored on the ledger. The details are retrieved using a HTTP post request to the application which sources the user’s information using the identifier the user used to log in. This request then returns a response with the user’s information which is stored in the ledger. This includes the users public certificate. As the certificate is over 900 characters long this is stored in an input box with the form. To view the certificate in the input box, the button “View prescription address” is clicked which displays the certificate in the input box along with a button to copy the full address easily this is seen in Figure 83. An example of the full details is seen in Figure 84.

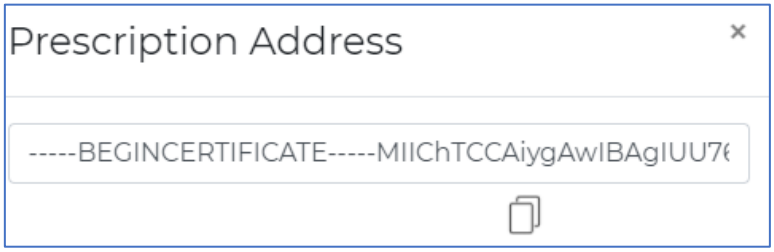



*Figure 83 - Prescription address*

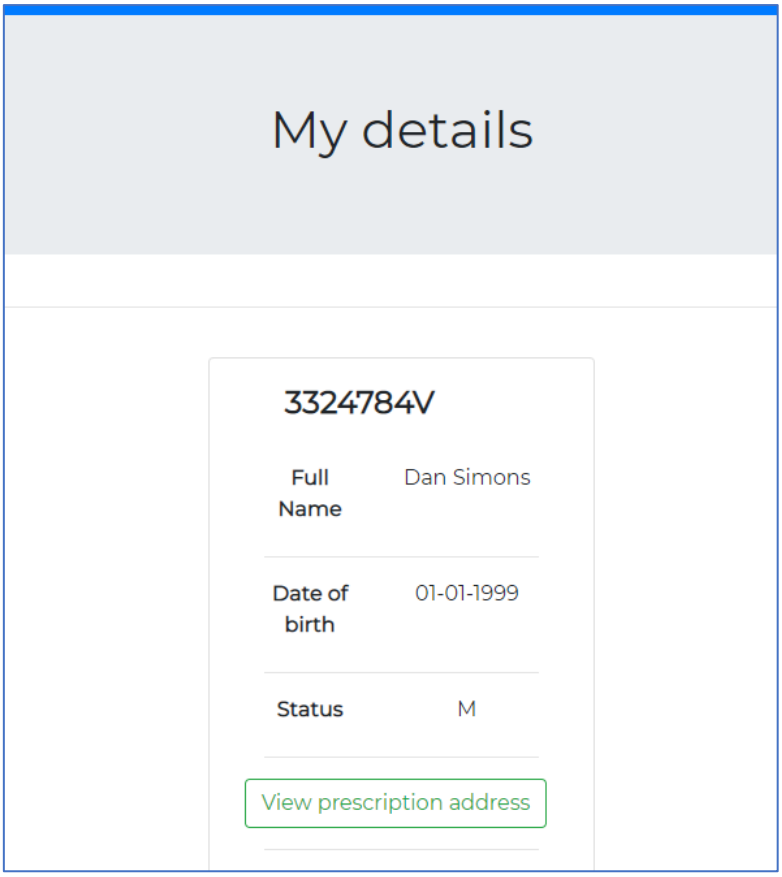



*Figure 84 - My details display*

### 4.7.4. Create Prescriptions

Creating a prescription is a core feature to the project that is only available to medical professionals. The restriction is implemented in the “NavBarUpdate.js” file which retrieves the status of the user currently logged in. There are 3 different types of status; “M” which stands for medical practitioner, “P” which stands for patient and “C” which stands for chemist. The “M” status allows access to creating a prescription on the website. If the user does not have the status “M” the link for “Create prescriptions” on the navigation bar is restricted and the URL is also restricted this is shown in Figure 85.

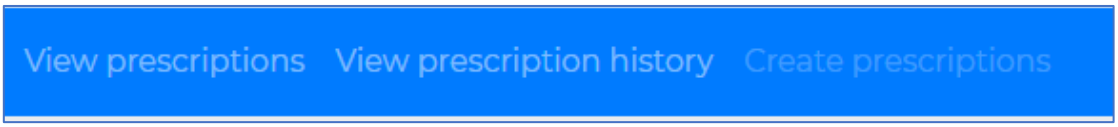



*Figure 85 - Restricted "Create prescriptions" on navigation bar*

When the user is authorized to access the “Create prescriptions” tab it gives them access to create prescriptions from a list of medication to select from. It also allows the user to add a comment tied to the prescription such as instructions in using the medication. In Figure 86 an example of the “Create prescription” webpage is shown.

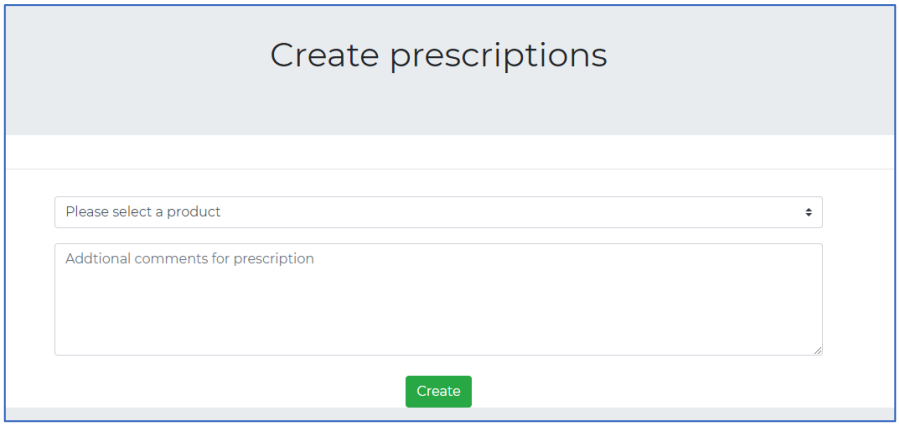



*Figure 86 - Create prescription webpage*

The “Create prescription” webpage functionality is powered by the “CreatePrescription.js” script which formats and sends the prescription data to the application. When the create button is pressed the prescription that was selected is sent to the application as a HTTP request where the application processes the data and submits it to the ledger. When this process is completed a notification appears which notifies the user that the prescription has been created this is seen in Figure 87.

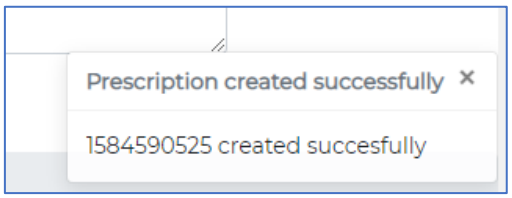



*Figure 87 - Create prescription notification*

This prescription is available to view in the “View prescription” tab. The prescription is first assigned to the user who created the prescription and then the user is free to transfer the prescription to a valid prescription address or in other words the certificate associated with that user.

### 4.7.5. View Prescriptions

The functionality to view prescriptions is another core element of the webpage. This feature allows users to see the prescriptions that currently they have ownership over. Initially when the page is loaded a HTTP post request is sent to the application to retrieve the prescriptions associated with the user currently logged in. These prescriptions are found by comparing the certificate of the user with the owner of each prescription. The application returns a list of prescriptions in Json format. After this data has been retrieved, each prescription found is displayed in a prescription widget. These prescription widgets are created with the “PrescriptionWidget.js” script which provides the functionality of creating the widgets, view the full prescription data and sending the prescription to another user. When the prescription data is first loaded the webpage displays each prescription with limited data of the “pID”, “date” and “productName” fields. Two buttons are displayed under the information which are “view prescription” and “transfer prescription”. An example of this is seen in Figure 88.

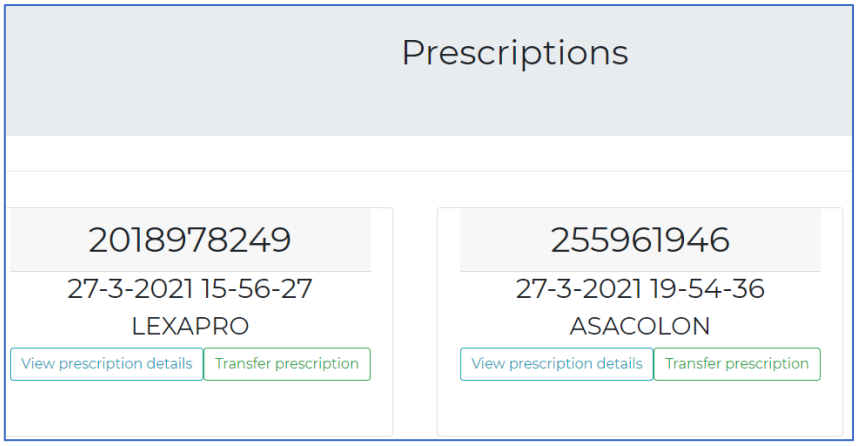



*Figure 88 - View prescriptions (Prescription widgets)*

When the "View prescription details" button is clicked, a dialog box appears with all the details of the selected prescription. Each attribute associated with a prescription is shown with the attribute name beside it. The prescription ID or PID associated with the prescription is displayed as the heading of the dialog box. These details display the owners and issuer's certificate that was used to sign the prescription with. A button is provided so that these two addresses can be copied easily. An example of the details found in a prescription is seen in Figure 89. The dialog box can be closed easily when the information is no longer needed.

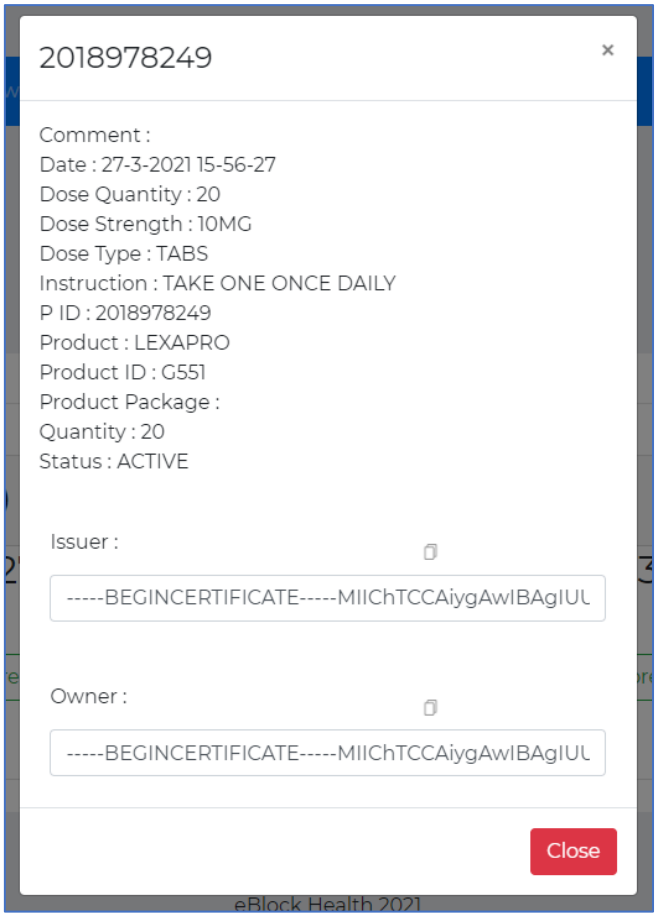



*Figure 89 - Prescription details*

### 4.7.6. Transfer Prescriptions

Another feature that was included in the "View prescription" page of the website was the ability to transfer or in other words send a prescription to another user on the network. This is to give the ability for a GP to send a prescription to a patient without the need for the patient to collect the prescription from the GP. The prescription is sent using the certificate associated with that user. In each prescription widget contains a button to transfer a prescription, this is shown in Figure 90.

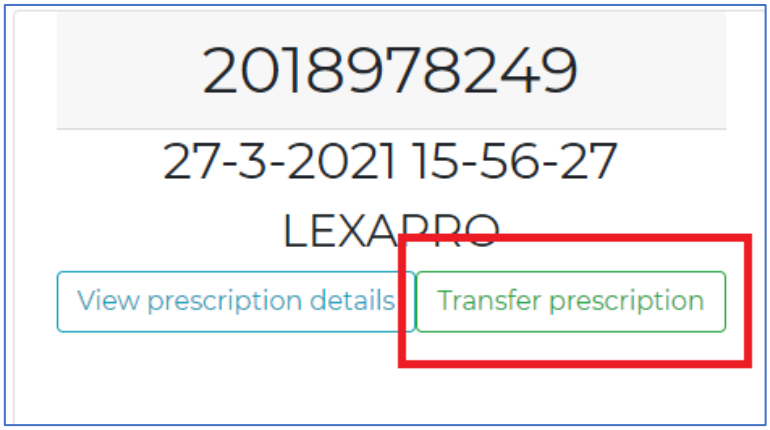



*Figure 90 - Transfer prescription button*

When the transfer prescription button is pressed a dialog window appears which asks the user to enter a user's certificate this dialog box is seen in Figure 91.

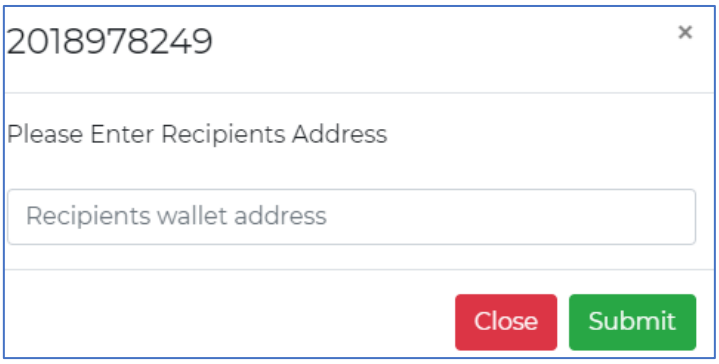



*Figure 91 - Transfer prescription dialog box*

This certificate, along with the PID of the prescription and the current owner of the prescription is sent with a HTTP post request to the appropriate method API call in the application. The certificate is verified to be a certificate of another user and the owner is checked to confirm the ownership of the prescription belongs to them. When this is complete the prescription owner is changed to the new owner and the confirmation is sent back to the presentation layer. A notification appears confirming the transaction and the list of prescriptions is updated to now show the prescriptions the user currently has and not the prescription that were transferred. If the transfer fails a notification that alerts the user of the failure appears. An example of the dialog box containing a prescription address and the notification after the prescription was successfully sent is seen in Figure 92.

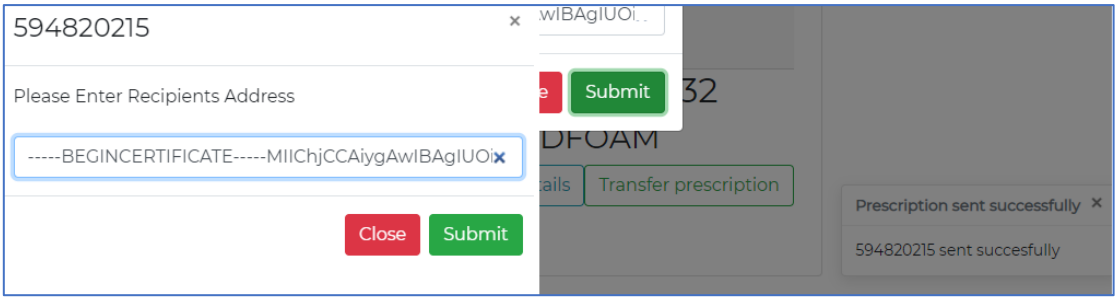



*Figure 92 - Successful transfer of prescription*

### 4.7.7. View Prescription History

The view prescription history functionality is a unique feature to the application which gives the ability to review the history of a prescription. For each prescription the complete history of the data of the prescription is stored on the ledger. This is due the fundamental fact that information on the blockchain cannot be deleted only updated in a new block. This was discussed in Section 2.5.1.

When the "View prescription history" webpage is loaded, the prescription history data of the prescriptions currently owned by the user is retrieved from the application layer with a HTTP post request. This request retrieves the complete history of the prescriptions and returns the data to the presentation layer. Similar to how the widgets were shown in the "view prescription" webpage, discussed in Section 4.7.5, widgets for each prescription retrieved are displayed on the webpage this is shown in Figure 95.

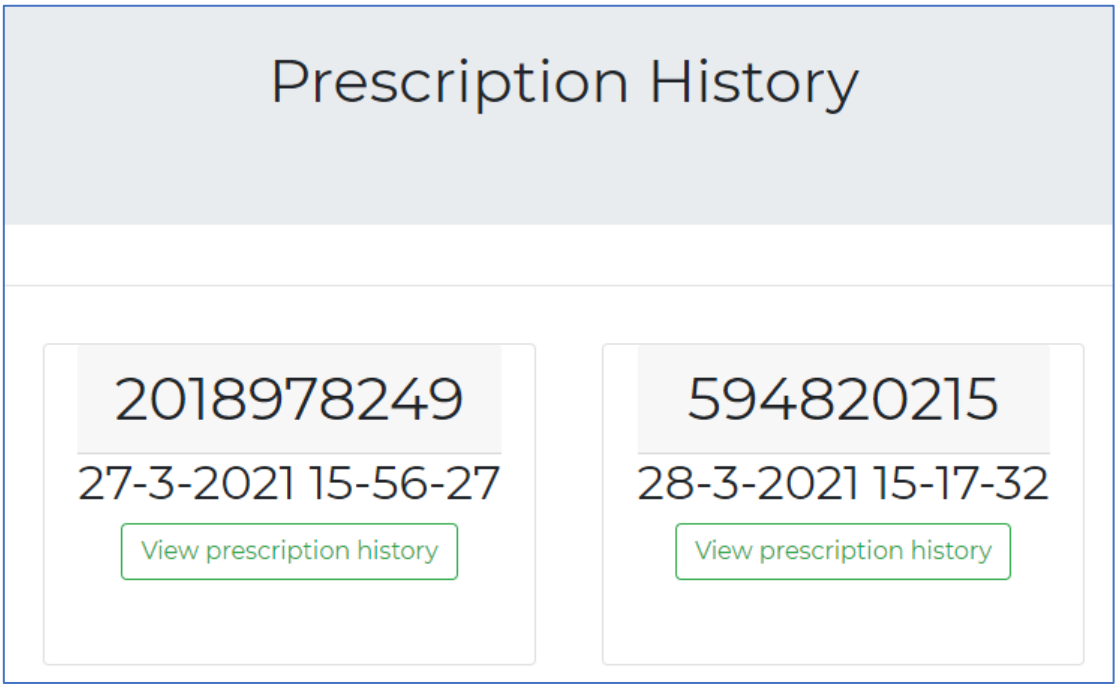



*Figure 93 - Prescription history widgets*

Each prescription history widget created, has a button which allows the user to click and view the prescription history of the widget. Once the button is clicked a table detailing the attributes of the prescription is created, along with a heading detail what prescription was

pressed and the owner and previous owner of the prescription. A pagination widget is also generated allowing the user to flick through the versions of the prescription and viewing the chain of ownership of the prescription. This feature allows each prescription to be traced to the owner proving its legitimacy and viewing any changes that could be made to the prescription. This feature is seen in Figure 96 where the history of one widget is shown.

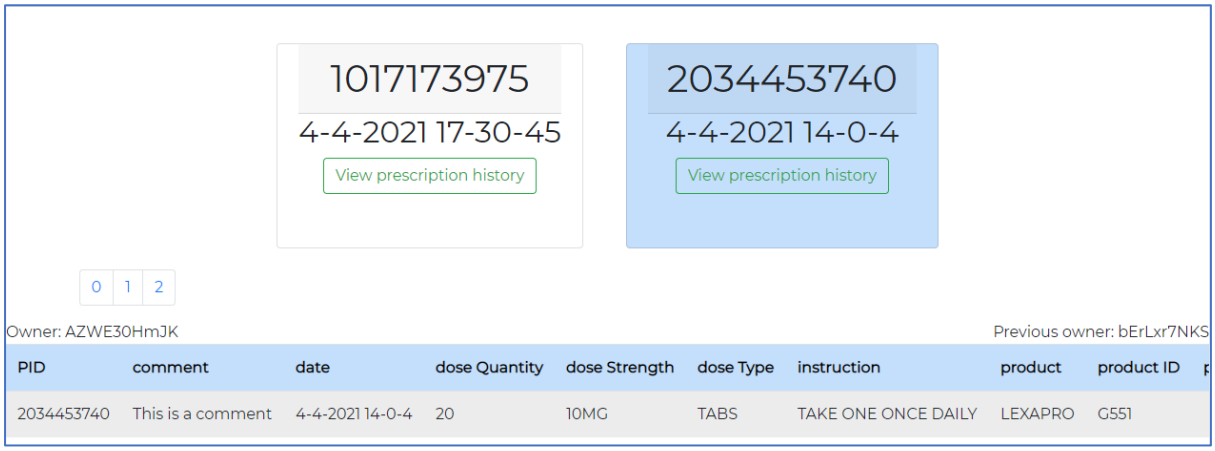



*Figure 94 - Prescription history display (multiple versions)*

The “Owner” and “Previous owner” fields in the Figure 93 represent the ownership of the prescription. These two fields display the certificates of the two users who owns and previously owned the prescription. Only a small selection of the certificate is displayed as each certificate is over 900 characters. This selection is 10 unique characters. A tooltip appears if the mouse hovers over the fields which displays the full certificate. If only one version of the prescription exists only the owner will be displayed along with only one tab in the pagination widget. An example of this is seen in Figure 95.


*Figure 95 - Prescription history (Single version)*

### 4.7.8. Conclusion

The prescription layer was planned and designed to be a user-friendly simple design with limited functionality to not confuse the user. Each of the core functionalities discussed in the design phase in Section 3.4.2 were implemented into the presentation layer of the project to give the ease of using a blockchain ledger without the need for technical knowledge. The use of bootstrap gave the website a standardized approach the development of the website. The use of JavaScript with HTTP request functionality ensured that up to date data was available in a quick and reliable manner while notifications provided insight to the process executing in the background to the user.

## 4.8. Challenges Encountered

This section discusses the many challenges that were encountered during the development phrase of the project. The source of the major challenges came with developing the data layer of the application as a lot of research was needed into the operation of the blockchain network. Some other challenges came through the configuration of the Tomcat server and the connection between each layer of the architecture.

### 4.8.1. Hyperledger Indy

Originally in this project, Hyperledger Indy was to be used to create and manage digital identities in a safe and secure manner. Hyperledger Indy enables the ability of self-sovereign identities. As discussed briefly in Section 2.4.6, self-sovereign identities give the user full control over who has access to their own digital identity. Indy shares these identities through peer to peer connection rather than storing it on a ledger.

Initially the documentation and the tutorials were followed, and a test network was created for the Hyperledger Indy, but the difficulties arose in trying to connect the Hyperledger Indy network with the Hyperledger Fabric. As Fabric uses a ledger to store information while Indy sends data through peer to peer connection a custom certificate authority in an organisation would have to be created to verify identities on the Hyperledger network. There is no current documentation that details the process involved with this and not external resources which provide answers. This process was halted to focus on the development of the full system.

### 4.8.2. Hyperledger Fabric Windows Deployment

Hyperledger Fabric was the framework used for the blockchain network and connecting the Java application to the network. It provided a decentralised ledger where data could be added to the ledger through methods in a smart contract. Hyperledger Fabric was discussed in Section 2.4.5.

Initially when configuring the Hyperledger Fabric, Windows was used as the operating system to run the network from. This caused countless issues with docker and the framework itself unable to execute commands. For a series of days, intense troubleshooting into the root of the problem led to reconfiguration of the network and docker itself but for every issue solved a new issue arose. The decision to create a virtual environment using a Linux distribution was taken, to remove the variables and the unknowns associated with running the network through Windows. Windows Subsystem for Linux (WSL) enables the easy use of a Linux distribution through Windows and the Linux operating system “Ubuntu 20.04” was chosen.

### 4.8.3. Smart Contract Chaincode

Deploying and creating the chaincode involved a lot of research and understanding into how it worked. Through this, countless issue arose that proved difficult to resolve. The first issue was the build automation tool for the Java chaincode application. Initially Maven was

chosen as the build automation tool for the chaincode application but because of the build structure from Maven the chaincode was incorrectly deployed to the network. This issue took a lot of investigation to discover as no clear errors in the deployment of the chaincode were ever give, it was only when the script “monitor.sh” was executed that the reason for deployment failure was discovered. The “monitor.sh” script was discussed in Section 4.5.2.3 and allows the logs of each of the docker containers in the network to be displayed. The build automation tool was changed to Gradle which resolved the chaincode deployment issue.

Another issue that stemmed from the deployment of the chaincode was the chaincode name. The name of the smart contract and the Jar file produced by the Java application had to be the same. This was another issue which took some time to resolve as the error which displayed through the “monitor.sh” script did not relate to what the actual error was. The error in the script pointed to a missing Jar file called “ShadowJar”. This issue once discovered was easily resolved by renaming the root directory of the Java smart contract application the same as the contract name specified in the contract.

### 4.8.4 Gradle war generation

As Gradle was necessary for the chaincode application, Gradle was also used for the Java web application as to mitigate the risk of issues occurring due to two sets of build automation tools. Although Gradle provides informative documentation, creating the build automation of a WAR file and deploying it to a Tomcat server proved to be slightly difficult. Gradle uses a plugin called “war” which creates a WAR file when the command line argument “gradle war” is used. Although this command generates a WAR file, the command does not specify where web content should be placed, and it does not generate a “WEB.XML” file automatically. Upon further research it was discovered that content placed in the directory “webapp” would be placed at the root of the WAR file. Using this information the script “warDeploy.sh” was created which automates the WAR file generation process which includes coping the content from the webapp folder into the Java web app and compiling into a WAR file. This WAR file is then deployed to the Tomcat server. This script is discussed previously in Section 4.5.2.5. A “WEB.XML” file was created and stored in the webapp folder.

### 4.8.5. Tomcat Install and Initialisation

Tomcat is a server which provides a Java web server environment to which WAR files are hosted and run.

The installation of Tomcat proved to have some difficulties. Tomcat 9, the most recent version of Tomcat had some build and install difficulties on the Ubuntu 20.04 which appear to be just this version. The Tomcat command line install does not contain all the content needed to run a Tomcat server from the Linux system. Due to this a previous version of Tomcat was installed, which was version 8.5. This was installed manually and proved to be effective, the only slight issue was that through this manual installation a daemon to start and stop the tomcat server was not created. Instead an environmental variable with the

location of the Tomcat directory was created which allowed the easy start and stop of the server manually. These commands were then implemented into the “warDeploy.sh” script.

Another issue that arose from Tomcat, was the restriction to HTTP post requests to the server. For a web application to send post requests they must first be enabled in the “web.xml” file where the parameter of “readonly” had to be set to false. This issue was quickly resolved but took time to discover as the error page gave no hints to what had caused the error in the first place.

## 4.9. Conclusions

This chapter discussed in detail the development process involved in this project. It started with outlining the software methodology, Agile Scrum, and the development sprint plan for this project. It then looked at the environment setup with GitHub and the Ubuntu virtual machine along with the different frameworks and programs needed to be installed. It then briefly discussed the technical architecture of the project. After this each layer of the architecture was discussed, the data layer of the application which discussed the blockchain network, the scripts needed and the chaincode developed for the layer. The application layer was then discussed which went through the different architectures, MVC and RESTful, along with process of connecting to the ledger and altering data on the ledger. Then the presentation layer of the project was detailed, which discussed the structure of web application and how each of the core functionalities operated in the system. Finally the challenges of the development process were discussed which outline what changes were made and how issues were fixed.

# 5. Testing and Evaluation

## 5.1. Introduction

Chapter 5 discusses the testing and evaluation that was implemented in this project. Section 5.2 will discuss the variety of tests used to evaluate the system against the requirements and specification of the system. Unit and systems testing will also be discussed and their results. White, black, and grey box testing will also be discussed and how it aided in the development and evaluating process. In section 5.3 the different types of evaluation will be discussed, which includes user acceptance testing of the system and the results from users who evaluated the system. The users were from a variety of different age groups and skill levels.

## 5.2. Testing

This section discusses the different types of testing used in the development of the Blockchain application. This includes black, white, and grey box testing which describes the types of testing used in parallel of the development process. Unit testing was implemented to test different components of the smart contracts that were developed. Finally, systems testing was used to test to functionality of the whole blockchain system and to ensure that each component integrated with each other as designed.

### 5.2.1. Black, White and Grey Box Testing

Black box testing is often referred to as functional testing and it is a testing technique which tests the project on the specifications of the project (88). In black box testing the software tester tests the code without knowledge of the internal source code or the internal mechanisms of the system. This means that the software tester focuses solely on the responses created by the system rather than how the response was created.

In this project two types of testing were implemented: systems testing and user acceptance testing. Both categories of evaluation are discussed in Section 5.2.3 and 5.3. respectively.

While black box tests the application from the point of view of the user, white box testing involves testing the internal structure of the code and to ensure the operations are performed according to specification of the project (88).

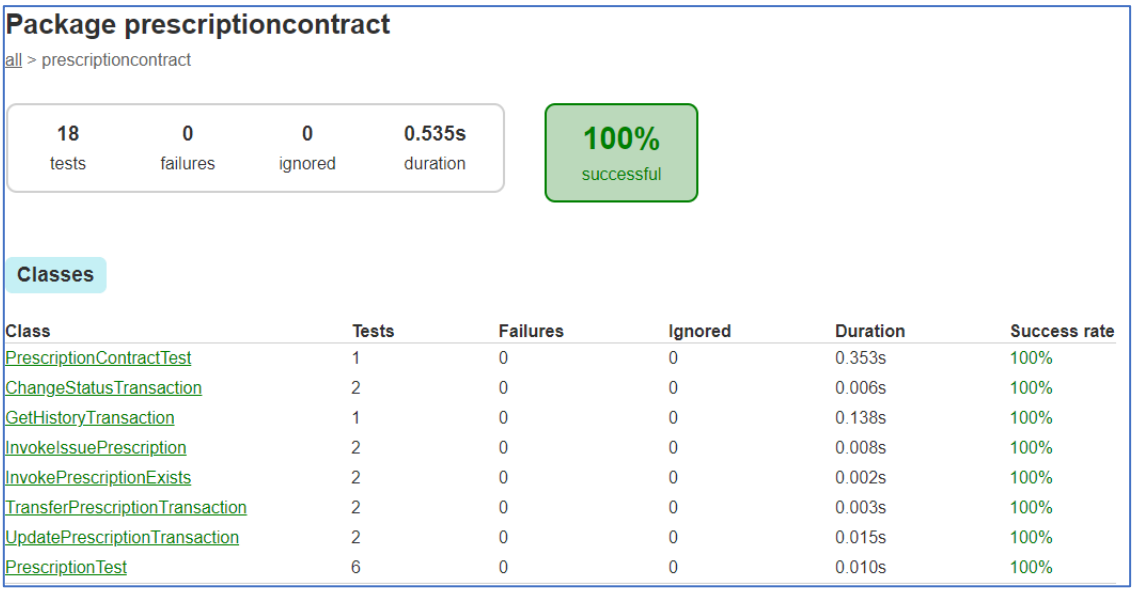
In this project, unit testing was used to test the internal structure of the code. These unit tests were implemented in each of the smart contracts and are discussed in Section 5.2.2.

Grey box testing is a test which combines white and black box testing. It is a technique which examines the software with partial knowledge of the application (89). Grey box testing was used by the developer to test features of the website and correct any issues that are encountered immediately such as incorrect REST calls to the application.

### 5.2.2. Unit Testing

Unit testing is a test whereby a single part of a component of the code is tested. Unit testing provides a method of verifying each component of a system and is used mainly in test-driven development (90). For this project, Junit 5 was used for the unit testing process (91). This provides an integrated approach to unit testing in Java and JVM environments.

Unit testing was used to evaluate the two smart contracts developed for this application. A total of 31-unit tests were developed. 18-unit tests were developed for the prescription contract while 13 were developed for user contract. Further tests were developed for the prescription contract as there was more methods to test. Each were tested to make sure that each transaction method could handle issues that might occur if an asset does or does not exist on the ledger. An example of the test output for each of the contracts is shown in Figure's 96 and 98.



**Package prescriptioncontract**

all > prescriptioncontract

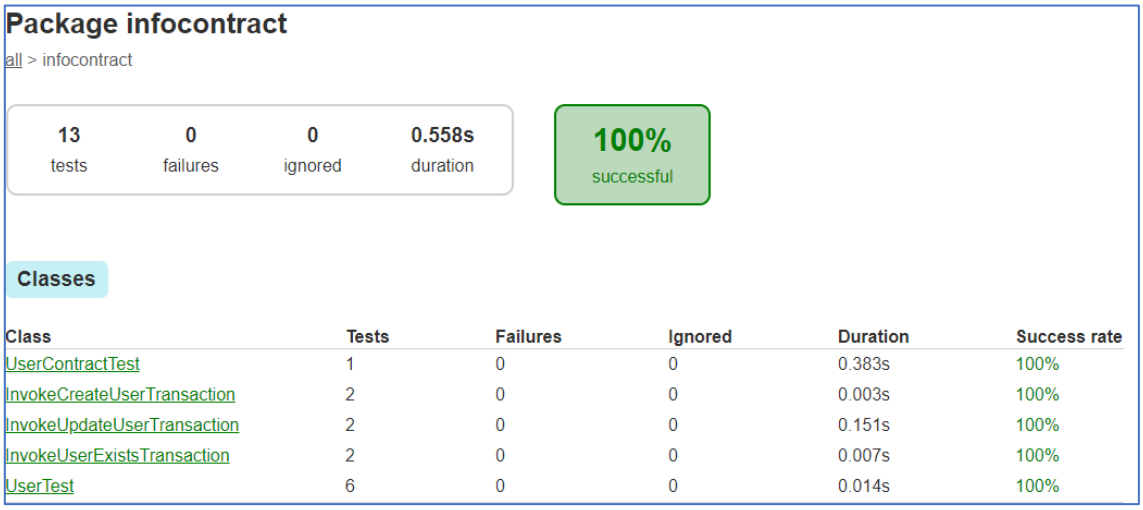
| 18 | 0 | 0 | 0.535s |
|---|---|---|---|
| tests | failures | ignored | duration |

100% successful

**Classes**

| Class | Tests | Failures | Ignored | Duration | Success rate |
|---|---|---|---|---|---|
| PrescriptionContractTest | 1 | 0 | 0 | 0.353s | 100% |
| ChangeStatusTransaction | 2 | 0 | 0 | 0.006s | 100% |
| GetHistoryTransaction | 1 | 0 | 0 | 0.138s | 100% |
| InvokeIssuePrescription | 2 | 0 | 0 | 0.008s | 100% |
| InvokePrescriptionExists | 2 | 0 | 0 | 0.002s | 100% |
| TransferPrescriptionTransaction | 2 | 0 | 0 | 0.003s | 100% |
| UpdatePrescriptionTransaction | 2 | 0 | 0 | 0.015s | 100% |
| PrescriptionTest | 6 | 0 | 0 | 0.010s | 100% |

*Figure 96 - Prescription contract test results*



**Package infocontract**

all > infocontract

| 13 | 0 | 0 | 0.558s |
|---|---|---|---|
| tests | failures | ignored | duration |

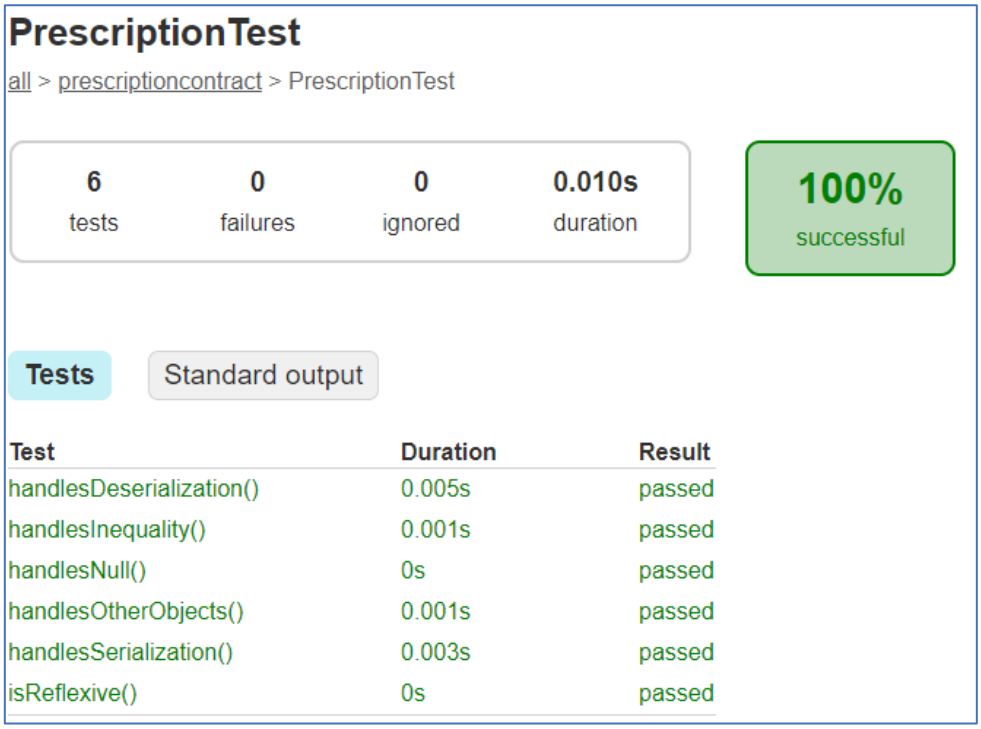
100% successful

**Classes**

| Class | Tests | Failures | Ignored | Duration | Success rate |
|---|---|---|---|---|---|
| UserContractTest | 1 | 0 | 0 | 0.383s | 100% |
| InvokeCreateUserTransaction | 2 | 0 | 0 | 0.003s | 100% |
| InvokeUpdateUserTransaction | 2 | 0 | 0 | 0.151s | 100% |
| InvokeUserExistsTransaction | 2 | 0 | 0 | 0.007s | 100% |
| UserTest | 6 | 0 | 0 | 0.014s | 100% |

*Figure 97 - User contract test results*

There were also tests implemented in each of the entities developed. In the prescription contract, the entity prescription was tested while in the user contract the user entity was developed. These tests were used to validate how the entity would handle serialization and deserialization as well as how the entity handles comparisons between itself and other objects. The results of these tests can be seen in Figure's 98 and 99.



**PrescriptionTest**

all > prescriptioncontract > PrescriptionTest

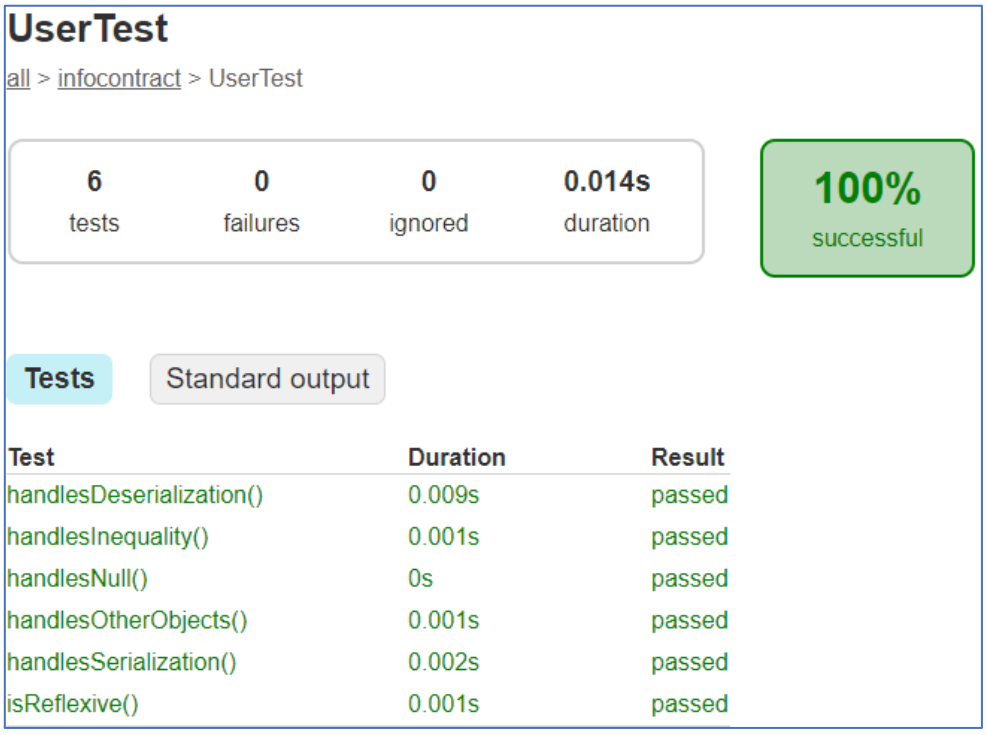
| 6 | 0 | 0 | 0.010s |
|---|---|---|---|
| tests | failures | ignored | duration |

100% successful

Tests | Standard output

| Test | Duration | Result |
|---|---|---|
| handlesDeserialization() | 0.005s | passed |
| handlesInequality() | 0.001s | passed |
| handlesNull() | 0s | passed |
| handlesOtherObjects() | 0.001s | passed |
| handlesSerialization() | 0.003s | passed |
| isReflexive() | 0s | passed |

*Figure 98 - Prescription entity test*



**UserTest**

all > infocontract > UserTest

| 6 | 0 | 0 | 0.014s |
|---|---|---|---|
| tests | failures | ignored | duration |

100% successful

Tests | Standard output

| Test | Duration | Result |
|---|---|---|
| handlesDeserialization() | 0.009s | passed |
| handlesInequality() | 0.001s | passed |
| handlesNull() | 0s | passed |
| handlesOtherObjects() | 0.001s | passed |
| handlesSerialization() | 0.002s | passed |
| isReflexive() | 0.001s | passed |

*Figure 99 - User entity test*

Unit tests help identity issues before the issue occurs and helps to investigate how each component of the software will operate. When the prescription entity was tested issues were discovered with the deserialization which were then resolved easily. This could have caused major issues later in the development if the issue was not fixed. Overall, unit testing provided a good insight into how each of the smart contracts worked with the system.

### 5.2.3. Systems Testing

Systems testing involves testing the whole project and validating that it meets the specification of the project. It usually involves a series of tests for each of the requirements of the system. Each system test was carried out by the developer as shown in Table 13.

| Test No. | Test Description | Expected Outcome | Result |
|---|---|---|---|
| 1 | Do new prescriptions add an entry to the blockchain? | A new prescription should be created from a transaction process on the blockchain network. | PASS |
| 2 | Can a user send a prescription to another user? | A prescription should be sent to the correct user, the user should receive the prescription. | PASS |
| 3 | Can a user receive a prescription from another user? | A user should be able to receive a prescription and have full access of the prescription. | PASS |
| 4 | Can a user send a prescription to a user that does not exist? | An error should notify the user that the user does not exist. | PASS |
| 5 | Can a user send a prescription to another user and still have access to the prescription? | A user should be able to send a prescription, they should no longer have ownership of that prescription. | PASS |
| 6 | When a user receives a prescription do, they have access to the prescription? | The user should have full access to the prescription they received. | PASS |
| 7 | Does the blockchain provide a clear immutable history of the transactions that have occurred? | The blockchain network should provide a clear history of the transactions that have taken place. | PASS |
| 8 | Can unauthorized users access other users details or prescriptions? | Only the individual who has authorized access to the prescription and details can view them. | PASS |
| 9 | Can a user log in with correct credentials? | The user should accept the user's details and log them into homepage. | PASS |
| 10 | Can a user log in with invalid details? | Website should prompt log in again and an error message. | PASS |
| 11 | Can a user successfully logout? | User logouts and returns to login screen. | PASS |

| | | | |
|---|---|---|---|
| 12 | Do all buttons and pages function on the webpage? | All links and buttons should give a response to the user. | PASS |
| 13 | Can a user view their details? | A user should be able to view their details. | PASS |
| 14 | Do prescriptions that are currently owned by the user display in the prescriptions tab in the website? | Only the current prescriptions that are owned by the user should be displayed in this website. | PASS |
| 15 | Can prescription history be viewed? | The full prescription history should be able to view, this includes the previous owner. | PASS |
| 16 | Do only users who are verified medical practitioners have the ability to create prescriptions? | Only medical practitioners should have access to the create prescription section. | PASS |

*Table 13 - System testing*

## 5.3. System Evaluation

Section 5.3 discusses the evaluation process of this project and the findings and comments received from the user acceptance tests carried out. Firstly it explains how the test were carried out and the results of the tests. Subsequently the comments and suggestions from the feedback are discussed.

### 5.3.1. User Acceptance Testing

User acceptance testing is a method of testing which the end user must verify that a system meets the requirements before the software can move to production. This was carried out by a variety of different end users, of different ages and different skill levels (92). 5 users were used to evaluate the system, where 60% of the users were male and 40% of the users were female. 40% were over the age of 35 and 40% of the participants described themselves as IT proficient.

Users were given the script outlined in Table 14.

| Test No. | Action | Expected Result | Actual Result (%PASS) |
|---|---|---|---|
| 1 | Login with Medical account (Details provided) | User logs in and is brought to home screen | 100% |
| 2 | Create prescription (Select any of the dropdown options) | User creates a prescription | 100% |
| 3 | View prescriptions in the view prescription tab | User view prescriptions and the newly created prescription should appear | 100% |

| | | | |
|---|---|---|---|
| 4 | View all details of prescription that was created | User should click the view details button on the prescription widget | 100% |
| 5 | View the history of the prescription that was created | User should click the view prescription widget and observe the details | 100% |
| 6 | View the details of the user | User should click the profile found under the identity | 100% |
| 7 | Send prescription to another user using provided certificate | User should send the prescription using the send prescription button in the view prescriptions widget | 100% |
| 8 | Log out of Medical account | User should click the log out button and the website should return the user to the log in screen | 100% |
| 9 | Login to Patient's account (Details provided) | User should log into second account | 100% |
| 10 | Verify the prescription has been received | The user should view the prescriptions of the account and confirm that the prescription is there | 100% |
| 11 | View the current history of the prescription | The user should view the history in the view prescription history tab where new data will have appeared concerning the previous owner of the prescription | 100% |
| 12 | View the details of the user | The user should click on the view details link which should display the details of that user | 100% |
| 13 | Attempt to reach the create prescription tab | The user should not be able to access the create prescription page | 100% |
| 14 | Log out of Patients account | The user should log out of the account and the website should redirect the user to the login page | 100% |

*Table 14 - User acceptance test instructions/results*

### 5.3.2. Usability Evaluation

After the user acceptance test, a series of evaluation questions were asked to identify areas that could be improved or areas that were missing from the website. Users were asked to answer questions and give impartial responses to a google form. The questions and the results of this form are attached to Appendix A. This section discusses the main points of the feedback which many users raised attention to.

80% of the users expressed that they had some sort of difficulty with the transfer prescription feature, with only 20% of users expressing they had no difficulty with this feature. Some users expressed that having the ability to send a prescription using another field other than the certificate would be a good future improvement. One user expressed that an edit button for the prescriptions would be useful. These responses can be seen in Figure 100.

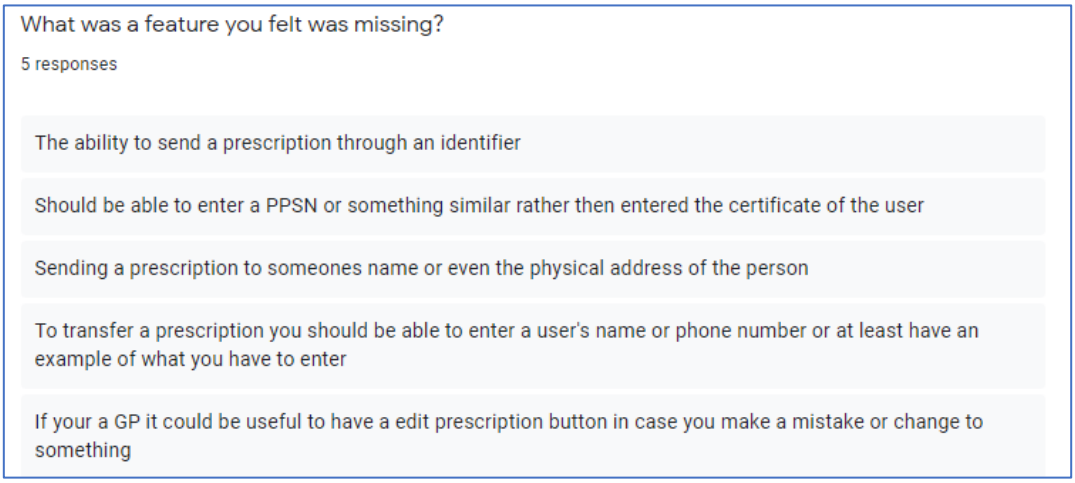



*Figure 100 - View prescriptions feedback*

Another concern raised by the users was found in the view prescriptions history page where 80% of users found the page confusing or unclear. The users expressed that they didn't quite understand what the use of the table was and that the previous owner field was confusing as the certificate didn't mean anything to them. Some users suggested to display the user's personal information rather than the certificate linked to the account. This feedback is shown in Figure 101.

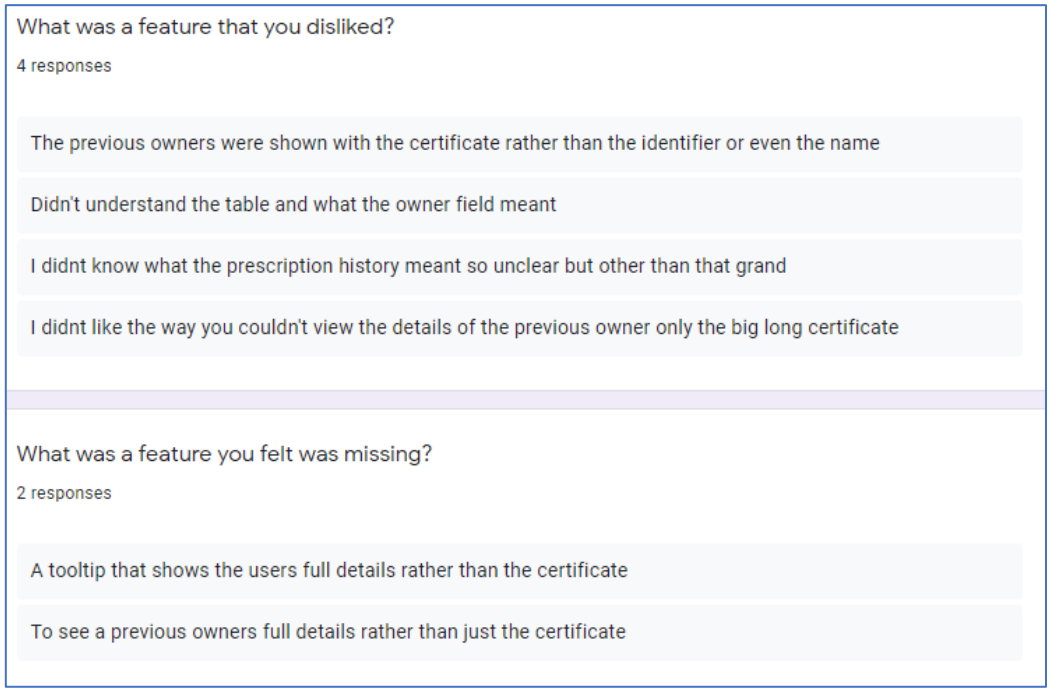



*Figure 101 - View prescription history feedback*

Some minor issues and suggestions brought up by some of the users included the ability to edit your own personal details and a better explanation of what the input fields require across the website. Some of the smaller improvements were implemented when the feedback was received while the bigger improvements were noted in the future work section (c.f. 6.6).

There was also a lot of positive feedback which was encouraging. The webpage received over 70% positive outlook on the design with many users stating that the webpage was simplistic or easy to use. Users stated that the use of widgets for prescriptions was a user-friendly design and it didn't overload them with information. Many users also commented on how the function to copy and paste the 900-character long certificates was a very useful feature. Excluding the transfer functionality, many of the users found each of the other core functionalities very easy to use which is again very promising. Finally, every user had a positive point on the tutorials on the home page of the website as they believed it had assisted them to navigate the system and provided an informative tutorial on how the website works.

## 5.4 Conclusion

This chapter reviewed the testing and evaluation of the system. This testing included unit testing which tested each method of each of the smart contracts deployed the blockchain network. These unit tests provided an extensive test on the functionality on each method of the chaincode. Systems testing was used to examine the webpage and to verify that it met the core requirements planned in the design phase of this project. The evaluation included user acceptance testing which users followed a test plan and then answered feedback about the webpage. This feedback provided insight into what could be improved with the project.

# 6. Conclusions and Future Work

## 6.1. Introduction

This project aimed to develop a system that tackles some of the issues with current prescription systems and to develop an ePrescribing system that was easy to use but also backed by the security of blockchain technology. Although the implemented application works well it is not ready for deployment to the real world. This project provides a proof of concept that blockchain infrastructure can be used with web-based technologies to create ePrescribing and ePrescription systems.

In this chapter the key lessons learned from this project will be discussed. In Section 6.2 a synopsis of the research conducted is discussed. In Section 6.3 the design process is discussed which includes the challenges that were faced during the design process. In Section 6.4 the issues that were faced in the development process are discussed and the choices that were made in the development. In Section 6.5 the difficulties of the evaluation process are discussed and finally, the future improvements and further areas of research that could be taken are discussed in Section 6.6.

## 6.2. Literature Review

The research undertaken for this project was vast as the subject matter is both multifaceted and wide ranging. The complexity of both blockchain technology and the different ePrescription systems around the world meant that a significant period was spent reviewing the literature for this project. The author explored both the Irish and international ePrescribing and ePrescription systems which provided insight into how the ePrescription process works and what could be improved in each of these systems. The benefits and the gaps drawn from each of the ePrescribing systems provided comprehensive knowledge into how each of the systems could be improved and what worked well in each of the systems. Time was also spent researching each of the different technologies needed for this project with a wide variety of different technologies explored and discussed. Blockchain was discussed and researched in this chapter as it is a core component of this project. The benefits and drawbacks of blockchain integrated with healthcare were also discussed. Finally, laws and legislation surrounding prescribing in Ireland and surrounding data protection rights were researched and presented. This in-depth research surrounding all relevant topics and information provided details into what was necessary for this project.

## 6.3. System Design

The design of the blockchain prescription system proved to be a difficult task, the designs were iterative throughout the design and development process to address the presenting issues. This was due to the complexity of the Hyperledger Fabric framework and the research needed into how the framework operated and created a blockchain network. Although documentation exists that provides information on how to build the network, there is limited information on integrating the network with a web application and integrating Hyperledger Indy with Hyperledger Fabric. The lack of information on the integration between the self-sovereign identities from Hyperledger Indy with the blockchain network of Hyperledger Fabric meant it was withdrawn from the implementation of the application. The removal of Hyperledger Indy from the project ensured that time was focused on the functionality of the exchange and creation of prescriptions through the Hyperledger Fabric network. The three-tiered architecture design ensured that the project was separated into three main components with their own responsibilities. This separation of concerns ensured that design flaws or issues could be traced easily to the source. This use of tiers integrated well with the Scrum agile methodology as each tier had their own respective sprints. The intensive planning and design phase meant that development of the Blockchain application followed a structure where each tier had an associated design.

## 6.4. System Development

During the development many issues were discovered that caused delays in the process. As mentioned in Section 6.3, the task of integrating Hyperledger Indy with Hyperledger Fabric proved difficult as no documentation existed between the two frameworks. This meant another smart contract had to be developed so that prescriptions could be sent to a specified user on the network.

Many other issues associated with the Hyperledger Fabric were present such as deployment issues with chaincode to the network. Another issue was discovered at the start of development process with the initialisation of the environment, this was due to the operating system (Windows), which had huge issues running some components of the network. Despite the issues encountered through the framework, every issue that was resolved led to a greater understanding of the development of the network.

As the project was built from the network up and split into tiers, the project was understood fully which meant that issues that were found during the development could easily be sourced to their respective component. Dividing the requirements and components into sprints meant that developing the project could be structured efficiently, requirements that could not be finished in the allotted time were returned to when more time was available.

## 6.5. Evaluation

Due to the restrictions of Covid-19 and the unfortunate fact that the developed application is only available locally to the authors workspace environment this meant evaluation of the system was restricted and could not receive an evaluation from a GP or pharmacist. Although this application was not evaluated by a healthcare professional, the system requirements and specifications were underpinned by the research conducted in chapter 2 which gave recommendations into what an ePrescribing system should contain. These requirements were the focus of the application and many forms of testing such as unit tests and systems test were carried out to ensure that these requirements were met. Although the security of the application was not fully tested, the underlying technology of Hyperledger Fabric provides sufficient security as Hyperledger Fabric is a private blockchain. Hyperledger Fabric only allows users registered to the network to access that data. This technology provides assurances that only users with a X.509 certificate registered with the network can access it. Evaluation was also carried out by several users which involved them following a test plan and then answering questions from a Google form. This evaluation process provided insight into what is needed in the system so it can be used by all ages and skill levels.

## 6.6. Future work

Through the development process several issues and improvements were discovered that could be implemented in a future improvement of this blockchain application to make it ready for deployment to the real world. In this section each improvement is discussed.

### 6.6.1. Securing Personal Data

As mentioned previously in Section 6.3 and Section 6.4, Hyperledger Indy which is used for self-sovereign identities could not be used due to lack of documentation regarding integration with Hyperledger Fabric. This meant that identities were saved to the blockchain ledger which meant personal data was kept on the ledger. This is a clear issue with GDPR and in future improvements this issue would be the first to be addressed. There are two possible solutions to this issue, the first would to be investigate how Hyperledger Indy could integrate with Hyperledger Fabric, this would require time and an extensive knowledge of how both frameworks operate. Another solution which would require less time would to be to integrate encryption and decryption of data stored on the ledger using the user's identity key information. The personal data associated with the user would be encrypted using the user's public key or certificate and then decrypted with their private key.

### 6.6.2. Identity Management

The next issue that would need to be addressed in future improvements of the application would be the identity management of the system; the X.509 certificates associated with each user that allows them access to the network. Currently this system saves each identity

to the server. This has both security, storage, and availability issues. Since the files contain the necessary information to access the network a breach of this would give an attacker access to the network.

Another issue would be if it was deployed to the internet a file for each user accessing the network would lead to possibly 1000s of files saved to the server which could overload the Tomcat server. Finally, as the files are saved to one server this would mean if the server were to go down, the identity information would not be accessible. A solution to this would be save identity information to a database hosted in each node of the network. This way identity information would be available if the network is active, the access would not be limited to one point like it is currently.

### 6.6.3. Additional Features

The next step in future development would be to create additional features that were not originally planned for the development of this project. Some of these features were highlighted through the user testing evaluation as discussed in Section 5.3.2. The ability to transfer prescriptions by the users identifier or even the address would be a necessary improvement to make this website accessible to all ages and skill levels. Alternative features that would build and improve the system are:

- The functionality to edit both the prescription and a user's personal details.
- The functionality to search for GP's or Pharmacies were information would otherwise be public, this way users could send prescriptions to their local pharmacy easily.
- The functionality for a GP to create a user's account so they could access the eBlock Health webpage. Currently users can only be created through the application and not the webpage.
- Finally the integration of a database which contains medicine products, this database would be up to date with the complete index of prescription medicines.

### 6.6.4. Further Testing

For future development of this application, further testing would be carried out which would test each component of the application and the security of it. Unit tests were only carried out on the chaincode of this application, but no unit tests were created for the Java web application or the presentation layer application. This would be necessary to test the full capabilities of each component in the system. Testing on the security of the application would be another necessary requirement of this application as sensitive data is stored on the ledger which would need to be protected at all costs. Stress testing the network to make sure it could handle a large number of requests would be another necessary test.

### 6.6.5. Deployment to Internet

Before deployment to the internet, the blockchain network would have to be altered to fit the needs and requirements of the system. For this project only 2 nodes were used as only the author was sending requests to the network. In order for this blockchain application to have the ability to handle large quantities of requests, it would be necessary for more network nodes to be a part of the network so if one network node fails the rest of the

network can handle the requests. Another requirement would be to initialise Secure Sockets Layer (SSL) HTTP requests with a recognized certificate authority. Although a SSL certificate was created for this application as Tomcat allows for SSL requests, most web browsers will not recognize the certificate created. Instead a certificate authority must be created to allow safe and secure transfer of data.

## 6.7. Conclusion

In conclusion, this project has been a tremendous learning experience. Researching into the area of blockchain and healthcare revealed a lot of information about these areas and also revealed that blockchain technology is still an emerging technology which has much promise. Developing the application proved to be a challenge but determined that blockchain technology could be utilized with other technologies to give an ePrescribing system. Although much more work would need to be completed on the project to make it production ready, this project is a proof of concept that blockchain technology can be used for other applications other than cryptocurrencies. This project proves that prescriptions can be hosted through web-based technologies while also relying on the secure backing of blockchain technologies.

# Bibliography

1.
National Standard for a Dispensing Note including a Clinical Document Architecture specificati on [Internet]. ; 2016 [cited 2021 March 29]. Available from: https://www.hiqa.ie/sites/default/files/2017-02/National-Standard-for-a-Dispensing-Note-including-a-Clinical-Document-Architecture-specification.pdf

2.
HEALTH INFORMATION AND QUALITY AUTHORITY. ePrescribing: An International Review [Internet]. ; 2018. Available from: https://www.hiqa.ie/sites/default/files/2018-05/ePrescribing-An-Intl-Review.pdf

3.
Pharmaceutical Society of Ireland - The Pharmacy Regulator [Internet]. Thepsi.ie. 2020 [cited 2021 Mar 11]. Available from: https://www.thepsi.ie/gns/home.aspx

4.
Guidance for prescribers and pharmacists on legislation changes to facilitate the safe supply of medicines during the COVID-19 pandemic PSI-The Pharmacy Regulator Medical Council Health Service Executive [Internet]. ; Available from: https://www.thepsi.ie/Libraries/COVID/Guidance_for_prescribers_and_pharmacists_on_legislation_changes_to_facilitate_the_safe_supply_of_medicines_during_the_COVID-19_pandemic.sflb.ashx

5.
Larkin J, Pericin I, Collins C. Healthmail Evaluation Report [Internet]. ; 2017 [cited 2021 Mar 7]. Available from: https://www.ehealthireland.ie/A2I-HIDs-Programme/Healthmail/Healthmail-Evaluation-Report-Final.pdf

6.
Li P, D. Nelson S, A. Malin B, Chen2 Y. DMMS: a Decentralized Blockchain Ledger for the Management of Medication Histories | Blockchain in Healthcare Today [Internet]. Blockchainhealthcaretoday.com. 2020 [cited 2021 Mar 9]. Available from: https://blockchainhealthcaretoday.com/index.php/journal/article/view/38/107#toc

7.
Larkin J, Pericin I, O'Mahoney B, Hull K, Collins C. An Evaluation of a Secure Email Service (Healthmail): A Cross-sectional Survey of Irish GPs. Universal Journal of Public Health [Internet]. 2018 Sep [cited 2021 Mar 7];6(5):284–97. Available from: http://www.hrpub.org/download/20180930/UJPH7-17611824.pdf

8.
General Data Protection Regulation (GDPR) – Final text neatly arranged [Internet]. General Data Protection Regulation (GDPR). 2019 [cited 2021 Mar 10]. Available from: https://gdpr-info.eu/

9.
McIntosh MJ, Morse JM. Situating and Constructing Diversity in Semi-Structured Interviews. Global Qualitative Nursing Research. 2015 Aug 14;2:233339361559767.

10.
Bates K, Beddy D, Whirisky C, Murphy M, O'Mahony JB, Mealy K. Determining the frequency of prescription errors in an Irish hospital. Irish Journal of Medical Science [Internet]. 2010 Feb 27 [cited 2021 Mar 28];179(2):183–6. Available from: https://link.springer.com/article/10.1007/s11845-010-0474-6#citeas

11.
MJF.ie. Healthmail Registration Portal [Internet]. Healthmail.ie. 2014 [cited 2021 Mar 28]. Available from: https://www.healthmail.ie/index.cfm

12.
Hull K. Monthly Statistics and Graphs for Healthmail, secure clinical email [Internet]. ; [cited 2021 Mar 28]. Available from: https://www.ehealthireland.ie/A2I-HIDs-Programme/Healthmail/Healthmail-Monthly-Report-June-2020.pdf

13.
Dierks T, Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.2 [Internet]. Www.hjp.at. 2020 [cited 2021 Mar 28]. Available from: https://www.hjp.at/doc/rfc/rfc5246.html

14.
KnowBe4. Phishing | What Is Phishing? [Internet]. Phishing.org. 2020 [cited 2021 Mar 1]. Available from: https://www.phishing.org/what-is-phishing

15.
Learning to detect phishing emails | Proceedings of the 16th international conference on World Wide Web [Internet]. Acm.org. 2013 [cited 2021 Mar 28]. Available from: https://dl.acm.org/doi/abs/10.1145/1242572.1242660

16.
Bhardwaj A, Sapra V, Kumar A, Kumar N, Arthi S. Why is phishing still successful? Computer Fraud & Security [Internet]. 2020 Sep [cited 2021 Mar 28];2020(9):15–9. Available from: https://www.sciencedirect.com/science/article/pii/S1361372320300981#bbib8

17.
PBS Information Management Section Pricing and PBS Policy Branch Technology Assessment and Access Division [Internet]. ; 2018 [cited 2021 Mar 29]. Available from: https://www.pbs.gov.au/statistics/expenditure-prescriptions/2018-2019/PBS_Expenditure_and_Prescriptions_Report_1-July-2018_to_30-June-2019.pdf

18.
Aldughayfiq B, Sampalli S. Digital Health in Physicians' and Pharmacists' Office: A Comparative Study of e-Prescription Systems' Architecture and Digital Security in Eight Countries. OMICS: A Journal of Integrative Biology. 2020 Sep 15;

19.
Htat K, Williams P, Mccauley V, Htat K, Williams P. Future of Australia' s ETP: Script exchange, script vault or secure mobile alternative. 2016;52–9. Available from: https://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1196&context=ism

20.
X-Road® — e-Estonia [Internet]. e-Estonia. 2019 [cited 2021 Mar 29]. Available from: https://e-estonia.com/solutions/interoperability-services/x-road/

21.
Overview — Invest in Estonia [Internet]. Invest in Estonia. 2019 [cited 2021 Mar 29]. Available from: https://investinestonia.com/business-opportunities/blockchain/overview/#our-advantages

22.
Parv L, Kruus P, Mõtte K, Ross P. An evaluation of e-prescribing at a national level. Informatics for Health and Social Care [Internet]. 2014 Aug 12 [cited 2021 Mar 29];41(1):78–95. Available from: https://pubmed.ncbi.nlm.nih.gov/25115948/

23.
X-TEE FACTSHEET EE [Internet]. X-tee.ee. 2020 [cited 2021 Mar 29]. Available from: https://www.x-tee.ee/factsheets/EE/#eng

24.
About Python | Python Institute [Internet]. Pythoninstitute.org. 2017 [cited 2021 Mar 30]. Available from: https://pythoninstitute.org/what-is-python/

25.
What is Python? Executive Summary [Internet]. Python.org. Python.org; 2020 [cited 2021 Mar 30]. Available from: https://www.python.org/doc/essays/blurb/

26.
index | TIOBE - The Software Quality Company [Internet]. Tiobe.com. 2020 [cited 2021 Mar 30]. Available from: https://www.tiobe.com/tiobe-index/

27.
Srinath K. Python–The Fastest Growing Programming Language. International Research Journal of Engineering and Technology (IRJET). 2017 Dec 4;4(12)(354-7).

28.
History of Java - Javatpoint [Internet]. www.javatpoint.com. 2011 [cited 2021 Mar 30]. Available from: https://www.javatpoint.com/history-of-java

29.
Herawan Dwika P. What is MySQL: MySQL Explained For Beginners [Internet]. Hostinger Tutorials. Hostinger Tutorials; 2018 [cited 2021 Mar 30]. Available from: https://www.hostinger.com/tutorials/what-is-mysql

30.
MySQL :: MySQL Customers by Industry [Internet]. Mysql.com. 2020 [cited 2021 Mar 30]. Available from: https://www.mysql.com/customers/industry/?id=

31.
DB-Engines Ranking [Internet]. DB-Engines. 2020 [cited 2021 Mar 30]. Available from: https://db-engines.com/en/ranking

32.
About Us [Internet]. MongoDB. 2020 [cited 2021 Mar 30]. Available from: https://www.mongodb.com/company

33.
Asay M. Why MongoDB Is Popular [Internet]. MongoDB. MongoDB; 2013 [cited 2021 Mar 30]. Available from: https://www.mongodb.com/blog/post/why-mongodb-popular.

34.
Fabric W. Open, Proven, Enterprise-grade DLT [Internet]. ; Available from: https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_fabric_whitepaper.pdf

35.
Hyperledger Indy – Hyperledger [Internet]. Hyperledger. 2020 [cited 2021 Mar 30]. Available from: https://www.hyperledger.org/use/hyperledger-indy

36.
ABOUT HYPERLEDGER [Internet]. ; Available from: https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf

37.
dotCloud - About [Internet]. Archive.org. 2013 [cited 2021 Mar 30]. Available from: https://web.archive.org/web/20140702231323/https://www.dotcloud.com/about.html

38.
Docker overview [Internet]. Docker Documentation. 2020 [cited 2021 Mar 30]. Available from: https://docs.docker.com/get-started/overview/

39.
Why Docker? | Docker [Internet]. Docker. 2013 [cited 2021 Mar 30]. Available from: https://www.docker.com/why-docker

40.
Rungta K. What is Selenium? Introduction to Selenium Automation Testing [Internet]. Guru99.com. Guru99; 2020 [cited 2021 Mar 30]. Available from: https://www.guru99.com/introduction-to-selenium.html#9

41.
What is JSON wire protocol in selenium? [Internet]. Tutorialspoint.com. 2019 [cited 2021 Mar 30]. Available from: https://www.tutorialspoint.com/what-is-json-wire-protocol-in-selenium

42.
About JavaScript [Internet]. MDN Web Docs. 2020 [cited 2021 Mar 1]. Available from: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

43.
Brown K. JavaScript: How Did It Get So Popular? [Internet]. Codecademy News. Codecademy News; 2018 [cited 2021 Mar 1]. Available from: https://news.codecademy.com/javascript-history-popularity/

44.
Chapter 2 [Internet]. W3.org. 2020 [cited 2021 Mar 1]. Available from: https://www.w3.org/People/Raggett/book4/ch02.html

45.
A brief history of CSS until 2016 [Internet]. W3.org. 2016 [cited 2021 Mar 1]. Available from: https://www.w3.org/Style/CSS20/history.html

46.
Maven – History of Maven [Internet]. Apache.org. 2013 [cited 2021 Mar 29]. Available from: https://maven.apache.org/background/history-of-maven.html

47.
Maven – Introduction [Internet]. Apache.org. 2013 [cited 2021 Mar 29]. Available from: https://maven.apache.org/what-is-maven.html

48.
Gradle User Manual [Internet]. ; [cited 2021 Mar 29]. Available from: https://docs.gradle.org/current/userguide/userguide.pdf

49.
Apache Tomcat® - Heritage [Internet]. Apache.org. 2021 [cited 2021 Mar 29]. Available from: http://tomcat.apache.org/heritage.html

50.
Bitcoin. BlockChain Technology [Internet]. ; Available from: https://scet.berkeley.edu/wp-content/uploads/BlockchainPaper.pdf

51.
Zheng Z, Xie S, Dai H, Chen X, Wang H. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. 2017 IEEE International Congress on Big Data (BigData Congress) [Internet]. 2017 Jun [cited 2019 Oct 15]; Available from: https://ieeexplore.ieee.org/document/8029379

52.
Castro M, Liskov B. Practical byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems. 2002 Nov 1;20(4):398–461.

53.
Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System [Internet]. 2008. Available from: https://bitcoin.org/bitcoin.pdf

54.
Ethereum [Internet]. CoinGecko. CoinGecko; 2018 [cited 2021 Mar 1]. Available from: https://www.coingecko.com/en/coins/ethereum

55.
Ethereum Whitepaper | ethereum.org [Internet]. ethereum.org. 2020 [cited 2021 Mar 1]. Available from: https://ethereum.org/en/whitepaper/

56.
The Eth2 upgrades | ethereum.org [Internet]. ethereum.org. 2020 [cited 2021 Mar 1]. Available from: https://ethereum.org/en/eth2/

57.
Van Saberhagen N. CryptoNote v 2.0. 2013; Available from: https://web.getmonero.org/resources/research-lab/pubs/whitepaper_annotated.pdf

58.
Rivest RL, Shamir A, Tauman Y. How to Leak a Secret. Advances in Cryptology — ASIACRYPT 2001 [Internet]. 2001 [cited 2021 Mar 2];552–65. Available from: https://link.springer.com/chapter/10.1007%2F3-540-45682-1_32

59.
Reid F, Harrigan M. An Analysis of Anonymity in the Bitcoin System [Internet]. ; [cited 2021 Mar 2]. Available from: https://users.encs.concordia.ca/~clark/biblio/bitcoin/Reid%202011.pdf

60.
Hyperledger Architecture, Volume 1 [Internet]. ; Available from: https://www.hyperledger.org/wp-content/uploads/2017/08/HyperLedger_Arch_WG_Paper_1_Consensus.pdf

61.
Garzik J. Public versus Private Blockchains Part 2: Permissionless Blockchains White Paper BitFury Group in collaboration with [Internet]. ; 2015 [cited 2021 Mar 5]. Available from: https://bitfury.com/content/downloads/public-vs-private-pt2-1.pdf

62.
Michael del Castillo. Blockchain 50: Billion Dollar Babies. Forbes [Internet]. 2020 Dec 3 [cited 2021 Mar 5]; Available from: https://www.forbes.com/sites/michaeldelcastillo/2019/04/16/blockchain-50-billion-dollar-babies/#1fa1accf57cc

63.
Bell L, J Buchanan W, Cameron J, Lo O. Applications of blockchain within healthcare. Blockchain in healthcare today. 2018;1(1-7).

64.
Union OJ of the E. COMMISSION DELEGATED REGULATION (EU) 2016, editor. Supplementing Directive 2001/83/EC of the European Parliament and of the Council by Laying down Detailed Rules for the Safety Features Appearing on the Packaging of Medicinal Products for Human Use. Official Journal of the European Union. 2015;27().

65.
electronic Irish Statute Book (eISB) [Internet]. Irishstatutebook.ie. Office of the Attorney General; 2020 [cited 2021 Mar 6]. Available from: http://www.irishstatutebook.ie/eli/2003/si/540/made/en/print#

66.
What is GDPR, the EU's new data protection law? - GDPR.eu [Internet]. GDPR.eu. 2018 [cited 2021 Mar 6]. Available from: https://gdpr.eu/what-is-gdpr/

67.
What are the GDPR Fines? - GDPR.eu [Internet]. GDPR.eu. 2018 [cited 2021 Mar 6]. Available from: https://gdpr.eu/fines/

68.
Wirth C, Blockchain S. Privacy by BlockChain Design: A Blockchain-enabled GDPR-compliant Approach for Handling Personal Data. [cited 2021 Mar 6]; Available from: https://dl.eusset.eu/bitstream/20.500.12015/3159/1/blockchain2018_03.pdf

69.
What information should my prescription contain? [Internet]. ; Available from: https://www.hse.ie/eng/services/list/1/schemes/cbd/acchealthcareireland/bringing-your-prescription-to-ireland-from-another-eu-eea-member-state.pdf

70.
Byrne A. Identiphone [Dissertation]. [School of Computing Technological University Dublin]; 2019.

71.
Mahon N. A Blockchain Risk Mitigation Solution for the Pharmaceutical Supply Chain [Dissertation]. [School of Computing Technological University Dublin]; 2019.

72.
Rovce W. MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS [Internet]. ; 1970. Available from: https://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf

73.
Adenowo A. Software Engineering Methodologies: A Review of the Waterfall Model and Object-Oriented Approach. Adenowo B, editor. International Journal of Scientific & Engineering Research [Internet]. 2013 Jun;4(7). Available from: https://www.researchgate.net/profile/Adetokunbo_Adenowo/publication/344194737_Software_Engineering_Methodologies_A_Review_of_the_Waterfall_Model_and_Object-_Oriented_Approach/links/5f5a803292851c07895d2ce8/Software-Engineering-Methodologies-A-Review-of-the-Waterfall-Model-and-Object-Oriented-Approach.pdf

74.
Manifesto for Agile Software Development [Internet]. Agilemanifesto.org. 2020 [cited 2021 Mar 12]. Available from: http://agilemanifesto.org/

75.
Kumar G. Impact of Agile Methodology on Software Development Process. Kumar Bhatia P, editor. International Journal of Computer Technology and Electronics Engineering (IJCTEE). 2012 Aug;2(4).

76.
Schwaber K. SCRUM Development Process [Internet]. ; [cited 2021 Mar 12]. Available from: https://scrumorg-website-prod.s3.amazonaws.com/drupal/2016-09/Scrum%20OOPSLA%201995.pdf

77.
Beck K. Test-Driven Development By Example [Internet]. ; 2002 [cited 2021 Mar 13]. Available from: http://barbra-coco.dyndns.org/yuri/Kent_Beck_TDD.pdf

78.
Pančur M, Ciglarič M. Impact of test-driven development on productivity, code and tests: A controlled experiment. Information and Software Technology [Internet]. 2011 Jun [cited 2021 Mar 13];53(6):557–73. Available from: https://reader.elsevier.com/reader/sd/pii/S0950584911000346?token=D155806114B6264B4F2412B47B4FE25E6ACFB3CAB5CB893C7063616C977FB745816898169786A9B7413D4B9FCD098614

79.
Y. Lai R, Fu Chan K. METHOD AND APPARATUS FOR SECURELY INVOKING A REST API [Internet]. 2013. Available from: https://patentimages.storage.googleapis.com/a4/1e/a9/04d6be6fe96821/US8621598.pdf

80.
Hursch WL, Lopes CV. Separation of Concerns. 1995;

81.
James AL, Rayfield T. Web-Application Development Using the ModelNiewlController Design Pattern. IBM T J Watson Research Center. 2015;

82.
Bora A, Bezboruah T. A Comparative Investigation on Implementation of RESTful versus SOAP based Web Services. International Journal of Database Theory and Application. 2015;8(3):297–312.
83.
Principled design of the modern Web architecture | ACM Transactions on Internet Technology [Internet]. ACM Transactions on Internet Technology (TOIT). 2017 [cited 2021 Mar 29]. Available from: https://dl.acm.org/doi/10.1145/514183.514185

84.
Li Q, Chen Y-L. Entity-Relationship Diagram. Modeling and Analysis of Enterprise and Information Systems [Internet]. 2009 [cited 2021 Mar 16];125–39. Available from: https://link.springer.com/chapter/10.1007/978-3-540-89556-5_6

85.
craigloewen-msft. About Windows Subsystem for Linux [Internet]. Microsoft.com. 2020 [cited 2021 Mar 20]. Available from: https://docs.microsoft.com/en-us/windows/wsl/about

86.
Peers — hyperledger-fabricdocs main documentation [Internet]. Readthedocs.io. 2021 [cited 2021 Mar 20]. Available from: https://hyperledger-fabric.readthedocs.io/en/latest/peers/peers.html

87.
Smart Contracts and Chaincode — hyperledger-fabricdocs main documentation [Internet]. Readthedocs.io. 2021 [cited 2021 Mar 21]. Available from: https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html

88.
Nidhra S. Black Box and White Box Testing Techniques - A Literature Review. International Journal of Embedded Systems and Applications. 2012 Jun 30;2(2):29–50.

89.
Ehmer Khan M, Khan F. A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. International Journal of Advanced Computer Science and Applications,. 2012;3(6).

90.
Olan M. UNIT TESTING: TEST EARLY, TEST OFTEN*. Journal of Computing Sciences in Colleges. 2003;19(2)(319-28).

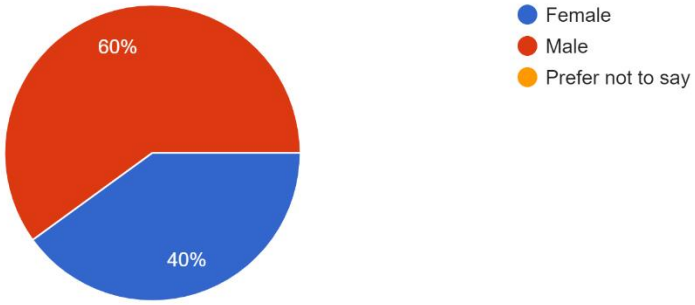
91.
JUnit 5 [Internet]. Junit.org. 2019 [cited 2021 Apr 1]. Available from: https://junit.org/junit5/

92.
Hambling B, Van Goethem P. User acceptance testing: a step-by-step guide. BCS Learning & Development; 2013.

# Appendix A

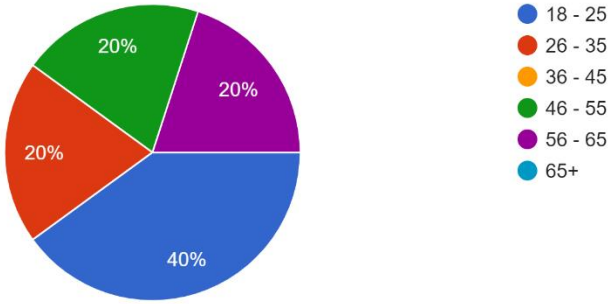
What is your gender?

5 responses



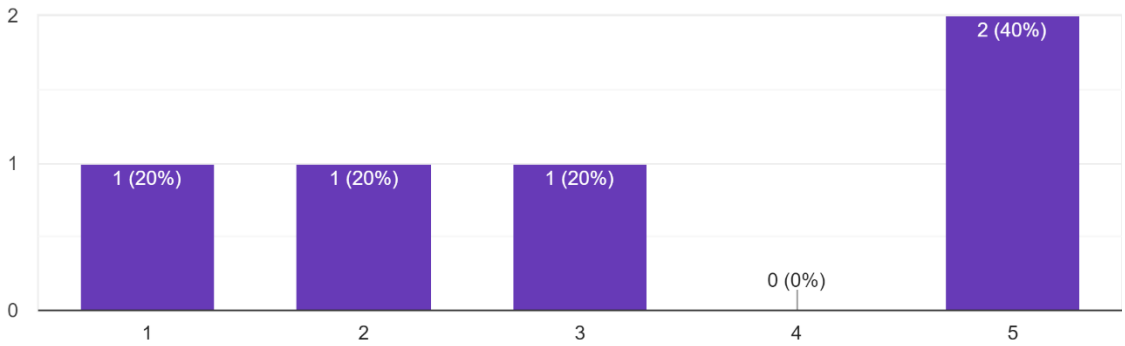

What age range represents you?

5 responses



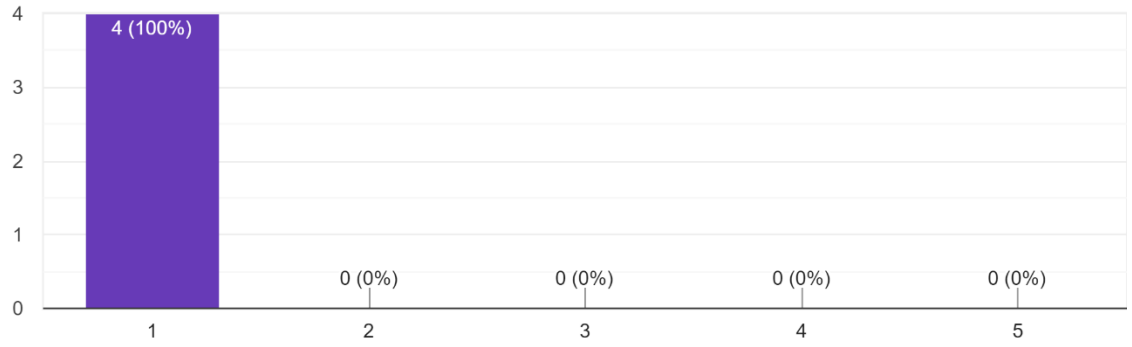

What represents your IT efficiency

5 responses



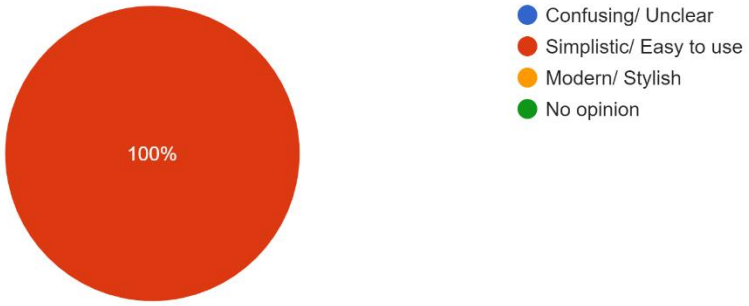
How difficult did you find the login process?

4 responses




What did you think of the design of the login page?

5 responses




What was a feature or a design that you liked on this page?

3 responses

The background of the login page looked cool

The ability to select the date of birth from the calendar tool

I liked the background design

What was a feature that you disliked?

1 response

The input for the passphrase dialog did not auto tab when you entered in a digit

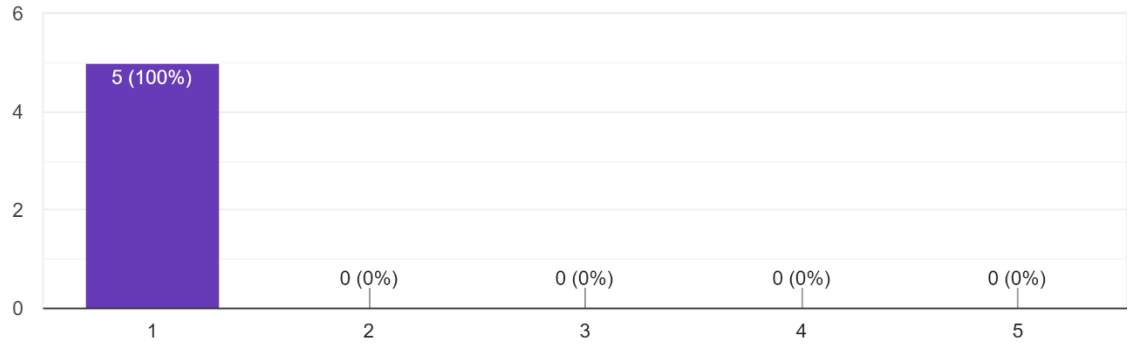
What was a feature you felt was missing?

1 response

Autotab when you enter a digit so that it goes to the next field

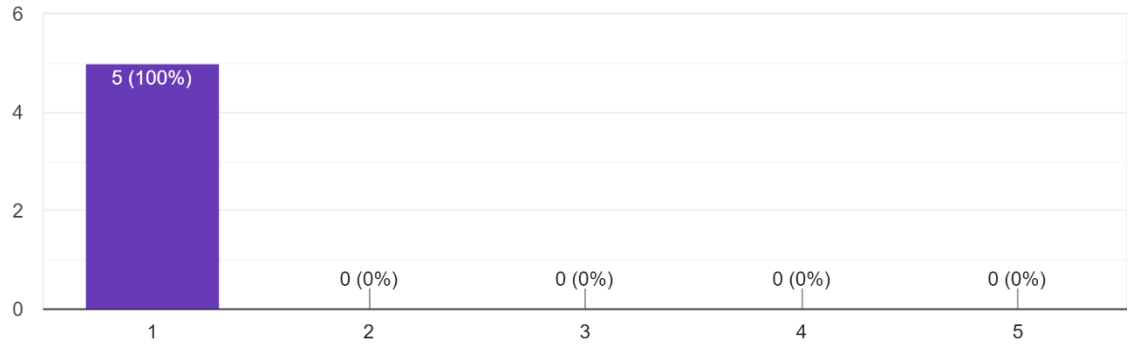
How difficult did you find to view prescriptions?

5 responses



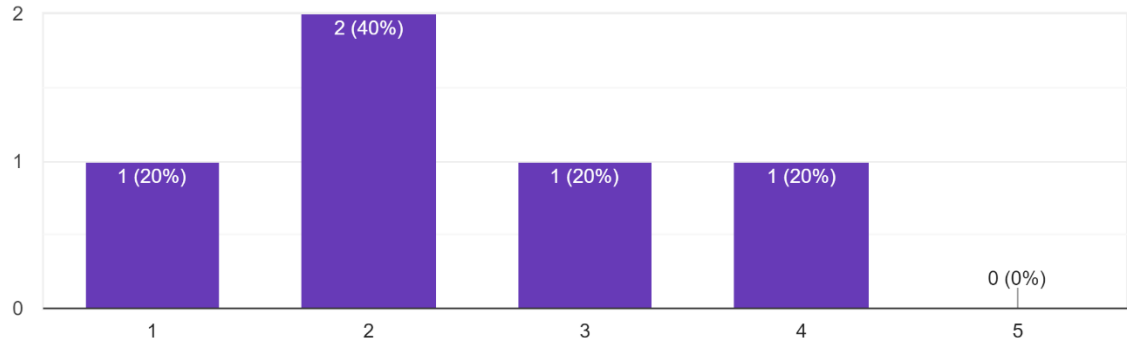

How difficult did you find to view the full details of a prescription?

5 responses



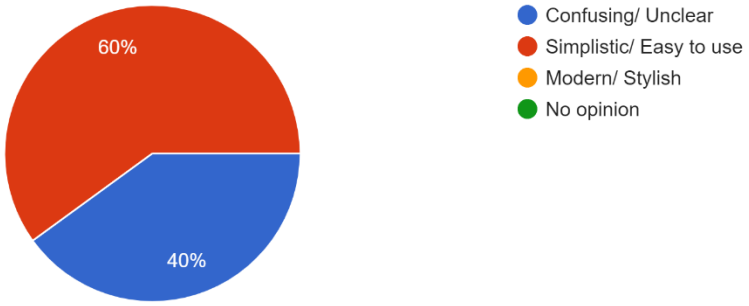
How difficult did you find to transfer a prescription to a user?

5 responses




What did you think of the design of the view prescriptions page?

5 responses




What was a feature or a design that you liked on this page?

3 responses

Each prescription was self contained in a widget and it only displayed the vital info which is much easier to read
The ability to copy-paste different certificates was also very handy

Each prescription only showed the information you need rather than all the info

A button to view the full prescription details was nice, so I could read it when I wanted

What was a feature that you disliked?

2 responses

You could only send a prescription with the users certificate

Sending a prescription was difficult as I didnt know what a certificate was, once I figured it out and typed it out it worked

What was a feature you felt was missing?

5 responses

The ability to send a prescription through an identifier

Should be able to enter a PPSN or something similar rather then entered the certificate of the user

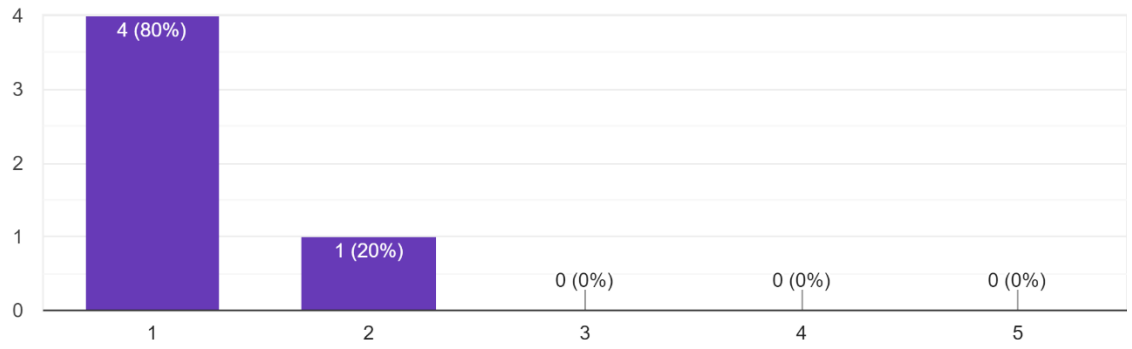
Sending a prescription to someones name or even the physical address of the person

To transfer a prescription you should be able to enter a user's name or phone number or at least have an example of what you have to enter

If your a GP it could be useful to have a edit prescription button in case you make a mistake or change to something

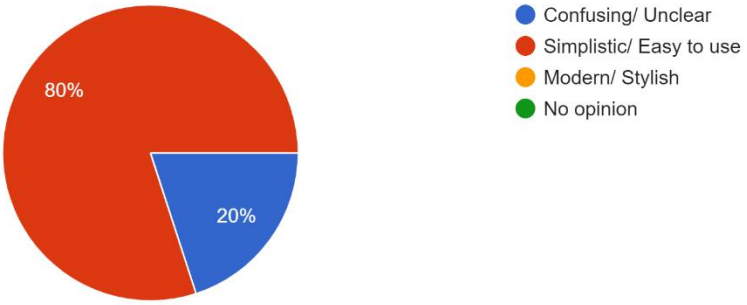
How difficult did you find create prescription?

5 responses



What did you think of the design for the create prescription webpage

5 responses




What was a feature or a design that you liked on this page?

3 responses

The drop down to select a prescription

The notification was handy as it told me that it worked which was nice

Very simple design, and extremely easy to understand even for me !

What was a feature that you disliked?

1 response

I didnt know what the text box was for

What was a feature you felt was missing?

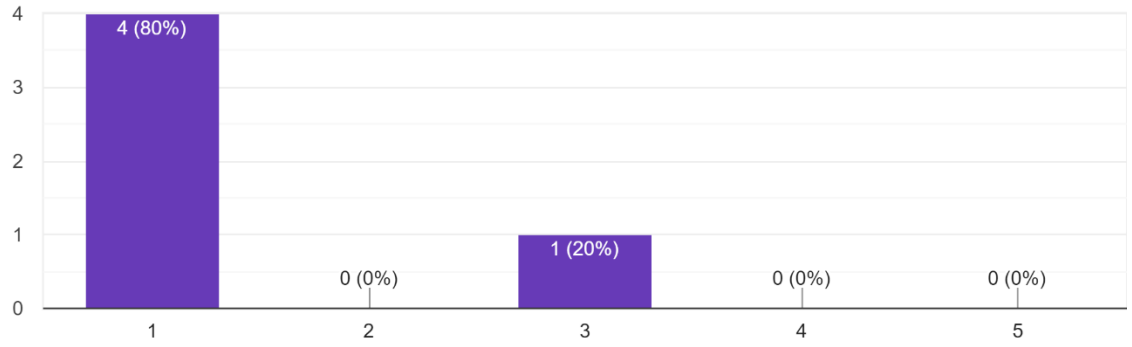
2 responses

I didnt know what the text box was for

Some description of what the text box is used for

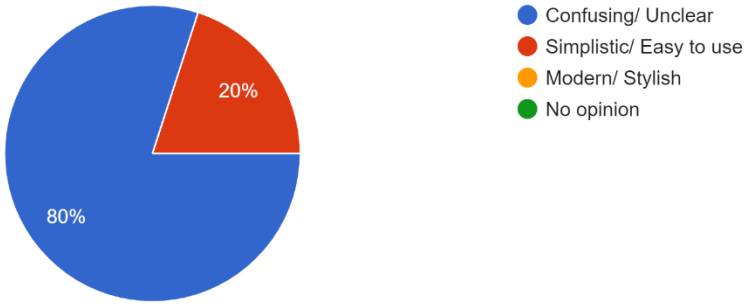
How difficult did you find to view prescription history?

5 responses




What did you think of the design of the view prescriptions history page?

5 responses




What was a feature or a design that you liked on this page?

2 responses

Similar to the view prescription tab, the self contained widgets were a nice touch

You could view previous owners of the prescriptions which was pretty cool

What was a feature that you disliked?

4 responses

The previous owners were shown with the certificate rather than the identifier or even the name

Didn't understand the table and what the owner field meant

I didnt know what the prescription history meant so unclear but other than that grand

I didnt like the way you couldn't view the details of the previous owner only the big long certificate

What was a feature you felt was missing?

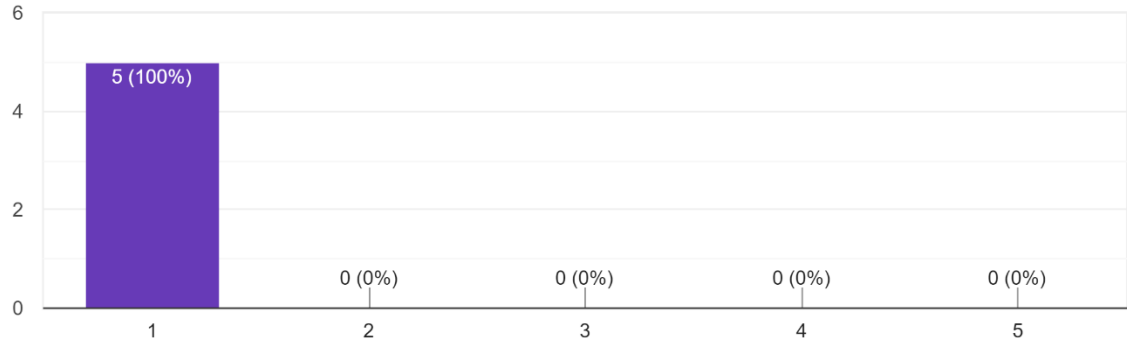
2 responses

A tooltip that shows the users full details rather than the certificate

To see a previous owners full details rather than just the certificate

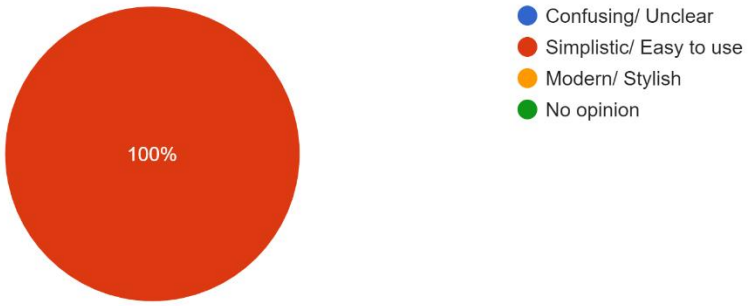
How difficult did you find to view a users personal details?

5 responses



What did you think of the design of the view user details page?

5 responses




What was a feature or a design that you liked on this page?

2 responses

The ability to copy and paste the cert easily with the copy button

Easy to follow and understand

What was a feature that you disliked?

0 responses

No responses yet for this question.

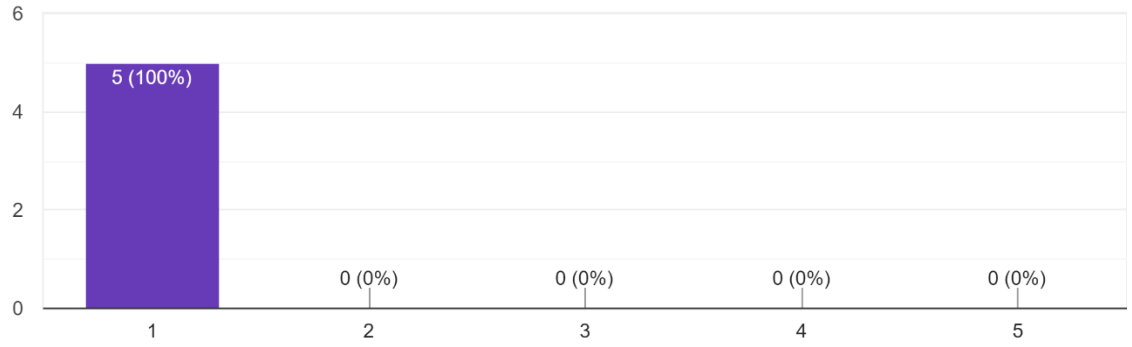
What was a feature you felt was missing?

1 response

The ability to edit your details

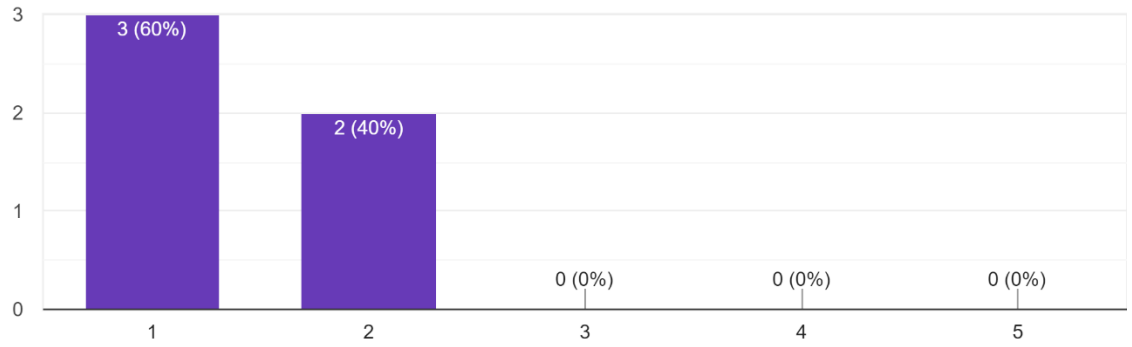
How difficult did you find it to navigate the navigation bar

5 responses



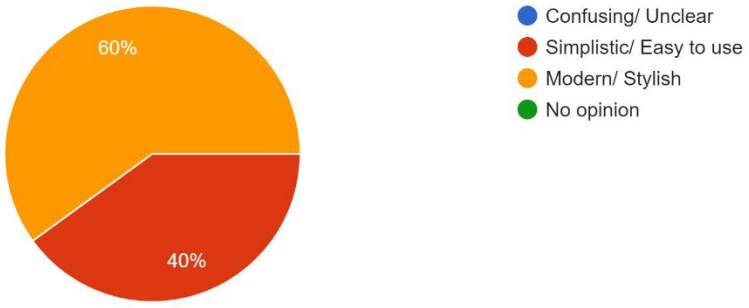

How difficult did you find it to log out of the website

5 responses




What did you think of the design of the home page?

5 responses



What was a feature or a design that you liked on this page?

5 responses

The tutorials for each of the features on the webpage

The tutorials that showed you how each feature worked

The tutorials were very helpful if figuring out how the website worked

The tutorials provided a lot of help when I found them guided through most things I had difficulty with

The tutorials and the design was very cool!

What was a feature that you disliked?

0 responses

No responses yet for this question.

What was a feature you felt was missing?

0 responses

No responses yet for this question.